Box–Cox–type Transformations for Linear and Logistic Models with Random Effects

Amani Mohammed Almohaimeed

A Thesis presented for the degree of
Doctor of Philosophy

Department of Mathematical Sciences
Durham University
United Kingdom

October 2018

Box–Cox–type Transformations for Linear and Logistic Models with Random Effects

Amani Mohammed Almohaimeed

Submitted for the degree of Doctor of Philosophy

October 2018

Abstract: Random effect models have become a mainstream statistical technique over the last decades; and the same can be said for response transformation models such as the Box–Cox transformation. The latter ensures that the assumptions of normality and of homoscedasticity of the response distribution are fulfilled, which are essential conditions for the use of a linear model or a linear mixed model. However, methodology for response transformation and *simultaneous* inclusion of random effects has been developed and implemented only scarcely, and is so far restricted to Gaussian random effects. The first aim of this thesis is to develop such methodology, thereby not requiring parametric assumptions on the distribution of the random effects. This is achieved by extending the "Nonparametric Maximum Likelihood" towards a "Nonparametric Profile Maximum Likelihood" (NPPML) technique. The implemented techniques allow to deal with overdispersion as well as two–level data scenarios in general linear models.

The second part of this thesis considers the transformation of mixed-effects

logistic models, with the aim of improving model fit. In binary data, link functions other than the logit can be used to connect predictors with the response. The Box-Cox transformation is used in mixed–effects binary regression models as an alternative link function for linearization purposes. The NPPML approach is used similarly as before, with some adjustments.

The proposed approach is implemented in the R package **boxcoxmix**. Simulation studies and applications on real data are carried out to study the performance of this approach.

Declaration

The work in this thesis is based on research carried out in the Department of Mathematical Sciences at Durham University. No part of this thesis has been submitted elsewhere for any degree or qualification.

Acknowledgements

First and foremost,

"Alhamdulillah for the countless blessings upon me."

My most sincere appreciation goes to my PhD supervisor Prof Jochen Einbeck. I am very grateful for his kindness, patience, encouragement, and immense knowledge throughout my PhD. I would like to extend my gratitude to all of the faculty and staff of the Department of Mathematics, Durham University.

My deepest and sincere gratitude goes to my husband and my parents for their love and encouragement.

Last but not least, I would like to thank my sponsors, Qassim University and Saudi Arabian Cultural Bureau in London for their generous scholarship and support.

Dedicated to

Bandar, Nareez, Albadra,
Mom and Dad

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In regression analysis, meeting the assumptions of normality and homoscedasticity of the response distribution and linearity of the model often requires transforming the response variable. The power transformation that was proposed by Box and Cox (1964) allows the response variable to achieve at least approximately a normal distribution, and makes the variance more nearly constant across data points around the regression line. Osborne (2010) suggested that normalizing data via the Box–Cox transformation to be a stage in data cleaning routines. In this thesis, we present the research in two parts. The methods of the first part focus on transforming the response in the linear model to validate the distributional assumptions of the model (Chapters 2 and 3). The second part applies the transformation to the odds–ratio as an alternative link function to generalize the logistic–mixed–type model and carry out the analysis of the binary response (Chapters 4 and 5). In this introduction, we initially lay out some basic concepts which are required for later use. A comprehensive review on the literature regarding these concepts will then be

presented. We will also review the statistical software packages used in this thesis. Finally, the chapter will be closed with a brief summary of this thesis.

1.1 Basic concepts and notations

The Box–Cox transformation (Box and Cox, 1964) has been widely used in applied data analysis. The objective of the transformation is to select an appropriate parameter λ which is then used to induce normality and homoscedasticity in the linear model. The transformation of the responses y_i, $i = 1, \ldots, n$, takes the form:

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_i & (\lambda = 0) \end{cases} \tag{1.1.1}$$

where the restriction $y_i > 0$ applies. The response variable transformed by the Box–Cox transformation is assumed to be linearly related to its covariates and the errors normally distributed with constant variance. For unknown λ,

$$Y^{(\lambda)} = X^T\beta + \epsilon \tag{1.1.2}$$

where $Y = (y_1, ..., y_n)^T$ is a vector of observations, $Y^{(\lambda)} = (y_1^{(\lambda)}, ..., y_n^{(\lambda)})^T$ is the vector of transformed observations, X is a known matrix of dimension $n \times p$, β is a $p \times 1$ vector of unknown predictors, $\epsilon \sim N(0, \sigma^2)$ is a vector of random errors. This family of transformations includes many traditional transformations to meet the needs of the data (Osborne, 2010):

$Y^{(1)}$: no transformation needed; produces results identical to original data

$Y^{(1/2)}$: square root transformation

$Y^{(1/3)}$: cube root transformation

$Y^{(1/4)}$: fourth root transformation

$Y^{(0)}$: natural log transformation

$Y^{(-1/2)}$: reciprocal square root transformation

$Y^{(-1)}$: inverse transformation and so on.

General ideas for finding variance-stabilising transformations, were discussed by Sakia (1992), are based on assuming that the variance of a Box-Cox transformed variable can be approximated by

$$Var(Y^{(\lambda)}) \simeq \sigma^2[E(Y)]^{2\lambda-2+\delta} \tag{1.1.3}$$

where δ is unknown and $[E(Y)] > 0$. If $\delta = 2 - 2\lambda$, the homoscedasticity is achieved.

In the linear model, it is assumed that a set of explanatory variables x_i, $i = 1, \ldots, n$, and a response variable y_i are linearly related such that $y_i = x_i^T\beta + \varepsilon_i$, where ε_i is an error term which is usually assumed to be Gaussian and homoscedastic. In such cases, the presence of further unknown variability can be accommodated by adding an unobserved random effect z_i with density $g(z)$ to the linear predictor,

$$y_i = x_i^T\beta + z_i + \varepsilon_i. \tag{1.1.4}$$

The responses y_i are assumed to be independently distributed with mean function $E(y_i|z_i) = x_i^T\beta + z_i$, conditionally on the random effect z_i. Let $\phi(y; \cdot, \cdot)$ denote the univariate Gaussian probability density function, with mean and variance specified in the remaining two function arguments. The conditional probability density function

of y_i given z_i is given by

$$f(y_i|z_i) = \phi(y_i; x_i^T\beta + z_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_i - x_i^T\beta - z_i)^2\right]. \qquad (1.1.5)$$

Note that under the presence of a random effect, the parametric intercept term can be omitted from $x_i^T\beta$. Under the non-parametric maximum likelihood (NPML) estimation approach, the distribution of the random effect will be approximated by a discrete distribution at mass–points $z_1, \ldots, z_K$, which can be considered as intercepts for the different unknown subgroups. The likelihood can now be approximated as a discrete distribution on a finite number K of mass-points z_k, with masses π_k (Aitkin et al., 2009)

$$L(\beta, \sigma^2, g) = \prod_{i=1}^{n} \int f(y_i|z_i)g(z_i)dz_i \approx \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_{ik} \qquad (1.1.6)$$

where $f_{ik} = f(y_i|z_k)$. For more details, see Chapter 2.

If the population from which the data are sampled consists of heterogeneous, unknown subpopulations, then the linear model described above will not fit well. The unobserved heterogeneity occurs when it is not possible to identify to which subpopulations the observations of a sample belong (Wang, 2004). Ignoring heterogeneity can result in biased and inconsistent estimates of all model parameters and the effect of covariates can also be meaningless (Assaf et al., 2016). Lubke and Muthén (2005) stated that "If the sources of heterogeneity are observed (e.g., gender), the data can be split into groups and the data analyzed with methods for multiple groups. If the sources of population heterogeneity are unobserved, the data can be analyzed with latent class models." For further information the reader is referred to Chapter 8 of the statistical modelling in R book (Aitkin et al., 2009).

The variance component model can be used to induce intra-class correlation in hierarchical two–level structures (e.g., children in school classes, hospital within region, etc). In this case, an unobserved random effect z_i with upper–level indexed by $i = 1 \ldots, r$, and lower-level indexed by $j = 1, \ldots, n_i$, $\sum_{i=1}^{r} n_i = n$ is added to the linear predictor $x_{ij}^T \beta$. The responses y_{ij} are independently distributed with conditional mean function $E(y_{ij}|z_i) = x_{ij}^T \beta + z_i$, where the distribution of the z_i is again unspecified. The conditional probability density function of y_{ij} given z_i is given by

$$f(y_{ij}|z_i) = \phi(y_{ij}; x_{ij}^T \beta + z_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_{ij} - x_{ij}^T \beta - z_i)^2\right]. \quad (1.1.7)$$

The likelihood is thus

$$L(\beta, \sigma^2, g) = \prod_{i=1}^{r} \int \left[\prod_{j=1}^{n_i} f(y_{ij}|z_i)\right] g(z_i) dz_i. \quad (1.1.8)$$

The aforementioned approach is carried out to approximate the likelihood in the same way as the random effect models with some related changes, for further details see Chapter 3.

A second topic considered in the thesis is the logistic model that includes an unobserved random effect z_i, $i = 1, \ldots, n$, with an unspecified mixing distribution $g(z)$ into the linear predictor. We assume the response Y_i, $i = 1, \ldots, n$, follows a binomial distribution with $Y_i \sim B(m_i, P_i)$ where m_i is the number of trials and P_i is a vector with fixed success probabilities for each category. Logistic models connect the probability P_i nonlinearly to the linear predictors $x_i^T \beta + z_i = \eta_i$ through a link function. In such case, a common link function is the logit that transforms the

interval $(0, 1)$ to $(-\infty, \infty)$,

$$\text{logit}(P_i) = \eta_i = x_i^T \beta + z_i \tag{1.1.9}$$

The likelihood function can be written as

$$L(\beta, g) = \prod_{i=1}^{n} \int f(y_i|P_i) g(z_i) dz_i \tag{1.1.10}$$

where

$$f(y_i|P_i) = \binom{m_i}{y_i} P_i^{y_i} (1 - P_i)^{m_i - y_i} \tag{1.1.11}$$

where $y_i = 0, 1, ..., m_i$, $P_i^{y_i}(1 - P_i)^{m_i - y_i}$ is the probability of having $m_i - y_i$ failures and y_i successes in a particular order, and $\binom{m_i}{y_i}$ is the binomial coefficient that is the number of ways of observing y_i successes in m_i trials. It follows $E(Y_i|z_i) = m_i P_i$ and $Var(Y_i|z_i) = m_i P_i(1 - P_i)$. If $m_i = 1$, then Y_i follows a Bernoulli distribution with mean and variance as $E(Y_i|z_i) = P_i$ and $Var(Y_i|z_i) = P_i(1 - P_i)$, respectively. Equation (1.1.9) is a binomial regression model that includes an unobserved random effect z_i, $i = 1, \ldots, n$, with an unspecified mixing distribution $g(z)$ and it includes the binary regression model as a special case. The model (1.1.9) is still sometimes referred to in the literature as a binary regression model and we will use these two terms interchangeably throughout the thesis. The marginal likelihood can again be approximated using NPML estimation (Aitkin et al., 2009):

$$L(\beta, z_1, ..., z_k, \pi_1, ..., \pi_k) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f(y_i|P_{ik}) \tag{1.1.12}$$

with a similar adoption for two–level models. See Chapters 4 and 5 for further information about fixed effect and mixed effects binary regression models.

1.2 Literature review

In regression analysis, the data needs to achieve normality and homoscedasticity of the response distribution in order to use parametric statistical tests. This often requires transforming the response variable. Box and Cox (1964) proposed a parametric power transformation technique for transforming the response in univariate linear models. They used maximum likelihood (ML) as well as Bayesian methods for the estimation of the transformation parameter. This transformation has been intensively studied by many researchers. Sakia (1992) briefly reviewed the work relating to this transformation. In this context, an estimation method for the Box–Cox transformation model without making any parametric assumption on the distribution of the error term has been proposed in Foster et al. (2001), Shin (2008) and Ji et al. (2017), and is found to be more consistent than the parametric estimation of the parameters. Further considerations relating to the estimation of the transformation parameter, particularly the non–consistency of the estimate, have also been discussed by Sugasawa and Kubokawa (2015), Maruo et al. (2015) and Maruo et al. (2017). They addressed the truncation problem that is the fact of not including the entire real line in the transformation due to the restriction on the response to be greater than zero. They concluded that the wrong assumption of the error term results in inconsistency of the ML estimation of the transformation parameters which might affect the inference of the other model parameters. Foster et al. (2001) has indicated that the non–parametric estimation of the transformation parameters is a useful tool for characterising the non–concave (or non–convex) shape of the transformation.

Solomon (1985) studied the application of the Box–Cox transformations to simple variance component models. The extension of the transformation to the linear mixed effects model was proposed by Gurka et al. (2006), in the case of a Gaussian random effect distribution. An obvious concern of assuming a normal random effect distribution is whether there are any harmful effects of misspecification. Agresti et al. (2004) showed that a misspecification of the random effects distribution may affect the prediction accuracy of the random effects as well as the fixed effects. In such cases, Maruo et al. (2017) whose interests were in assessing fixed effects more than random effects, added a robust inference to the model proposed by Gurka et al. (2006). They proposed an inference of the median difference procedure based on the Box–Cox linear mixed model by applying the inverse of the transformation to the model mean and obtain the corresponding median on the original scale. However, Carroll (1982) focused on estimating the median of the response in the original scale when the choice of the power of the transformation is restricted to a finite set and found that restricted estimation of the transformation parameter can possibly lead to inferences different from ML estimation of the median response. In consideration of the random effects misspecification, Wang et al. (2012) argued that even when the estimation of the fixed effect is robust, the estimation of the random effects could be invalid. This may raise the question of whether to use fixed or random effects in transforming such data. Clark and Linzer (2015) offered general rules of thumb upon which researchers may rely when deciding between the fixed effects and random effects approach. They concluded that, for any particular sample, the random effects model may introduce bias in estimates of the parameters but can greatly constrain the variance of those estimates leading to estimates that are closer

to the true value when compared with those of the fixed effects model.

Bock and Aitkin (1981) showed that there is no need to make an assumption about the distribution of the random effects and that is estimated as a discrete mixing distribution. Aitkin (1996a), Heckman and Singer (1984) and Davies (1987) showed that the parameter estimation is sensitive to the change in the mixing distribution specification. The problem of estimating the mixing distribution using a specific parametric form (e.g. normal) can be overcome by the use of non–parametric maximum likelihood (NPML) estimation; the NPML estimate of the mixing distribution is known to be a discrete distribution involving a finite number of mass–points and corresponding masses (Laird, 1978; Lindsay, 1983). Agresti et al. (2004) studied the effects of using parametric or nonparametric estimation methods for random effects when the true distribution is quite far from normal. They concluded that "the safest approach might seem to be always to use a nonparametric rather than a parametric approach for the random effects distribution." Methodology to assess the accuracy of random effect distributions has been developed by Verbeke and Molenberghs (2013).

Theory and application of the non–parametric approach to random effects are acknowledged to be "attractive" (Butler and Louis, 1992). Rabe-Hesketh et al. (2003) used NPML estimation to estimate logistic regression models models with measurement error, and their study revealed that the NPML gives unbiased estimates of the odds–ratio and other parameter of interest. The study by Fotouhi (2003) used the bias and efficiency criteria to compare five estimation procedures for fitting multilevel models. They recommended the NPML approach among the studied

procedures especially when no prior information is made about the distribution of the random effect. Butler and Louis (1992) concluded that, in both linear and non–linear models, the non–parametric approach produces efficient and robust estimates of fixed effects.

Maximization of a nonlinear likelihood function is required for estimation of the Box–Cox Equation (1.1.1) which has been remarked by Spitzer (1982) as a more complex analysis. Gurka et al. (2006) used the residual maximum likelihood (REML) to find the parameter estimators of the Box–Cox Equation (1.1.1) for the linear mixed effects model. Piepho and McCulloch (2004) noted that ML estimation of variance components is more biased than REML estimation. However, Aitkin (1995) demonstrated that the NPML method reduces bias and increases precision. Iterative methods must be used to find the parameter estimates that maximize the likelihood. Lindstrom and Bates (1988) compared the Newton–Raphson (NR) and Expectation–Maximization (EM) algorithms in terms of computational order and performance in estimating the parameters in the mixed-effects model via both ML and REML. They concluded that although the NR algorithm achieves convergence with a small number of iterations, it is not guaranteed to converge, while the EM algorithm will always converge to a local maximum of the likelihood but may require a high number of iterations.

The EM algorithm for NPML estimation was proposed by Laird (1978) in the case of mixture density estimation and developed by Lindsay (1983). This algorithm was used by Dempster et al. (1977) for fitting the finite mixture distribution, each iteration of this algorithm is based on two steps: the expectation step

(E-step) in which the posterior probabilities that the current unit is assigned to a certain cluster are computed, and the maximization step (M-step) in which the ML estimates are calulated using the current weights. The ML estimate via the EM algorithm is a preferable approach due to its generality and simplicity; when the underlying complete data come from an exponential family whose ML estimates are easily computed, then each maximization step of an EM algorithm is likewise easily computed (Dempster et al., 1977). For both overdispersed and variance component models, the EM algorithm for the NPML estimate of the mixing distribution was regarded as "very stable and converged in every case" (Aitkin, 1996a, 1999a). We refer the reader to Aitkin (1999a), Aitkin et al. (2009) and Einbeck et al. (2007) for more details in the context of linear models. A brief discussion on the EM algorithm for the NPML estimate of an unspecified mixing distribution for mixed–effects logistic models can be found in Trovato and Caiazza (2004).

A particular benefit of the NPML approach is that the posterior probability that a certain unit belongs to a certain cluster corresponds to the weights in the final iteration of the EM algorithm (Sofroniou et al., 2006). Another advantage of this approach is that there is no need for a computational effort to locate new mass–points when the number of components increased and that the mass–points are not necessarily restricted to be on a grid (Aitkin, 1996a). Aitkin concluded that "the simplicity and generality of the non–parametric model and the EM algorithm for full NPML estimation in overdispersed exponential family models make them powerful modelling tools".

Einbeck and Hinde (2006) investigated the effect of the number of com-

ponents K on convergence of EM algorithm, they concluded that a larger value of K may results in a large number of iterations for convergence. This implies that the value of K needs to be estimated. Aitkin (1999a) suggested that in order to select the appropriate number of components in the finite mixture one can start with one component (i.e. the standard generalized linear model) and then increase the number of components until the likelihood is maximized. A general common issue in clustering techniques is the difficulty of determining the 'right' number of components. Within the context of NPML estimation, Böhning et al. (2006) remarked that "profile likelihood ratios will not have standard χ^2–distributions", therefore, they suggested using other selection criteria for determining the number of components. Bowman and Evers (2017) showed that one cannot use classical statistical tests for estimating the number of components due to the parameter boundary hypothesis problem. To solve this, they suggested the use of model selection criteria. Leroux and Puterman (1992) indicated that the NPML estimation may require an unnecessarily high number of components to maximize the likelihood whereas well–fitting models with a small number of components are usually preferred. Lukociene and Vermunt (2009) suggested an approach in which the number of components is estimated. In their approach, the value of K increased until no further improvement is possible for the criterion used for model selection. Akaike's information criterion (AIC; Akaike, 1998) and the Bayesian information criterion (BIC; Bhat and Kumar, 2010) are popular information criteria for comparing model fits. The model with the 'correct' number of classes is the one with the minimum AIC or BIC value.

The ability of the EM algorithm to locate the global maximum in fewer iterations can be affected by the choice of initial values. Aitkin et al. (2005) demon-

strated that fitting mixture models using the EM algorithm guaranteed convergence to at least one local maximum, however, extensive search over the starting values was suggested to locate the global maximum. Einbeck and Hinde (2006) noted that the EM algorithm may find different local maxima, depending on the choice of the starting values. The difference in the local maxima may occur depending on whether the EM algorithm has an odd or even number of mass–points (Aitkin, 1996a). Several methods for choosing initial values for the EM algorithm in the case of finite mixtures are discussed by Karlis and Xekalaki (2003). A grid search for setting the initial values was suggested by Laird (1978).

Hou et al. (2011) compared the effect of estimating the Box-Cox power transformation parameter and subsequent analysis of variance with or without a priori knowledge of predictor variables under the fixed effect or random effects model cases. They found limited difference from subsequent test of structural effects regardless of whether such structure is included or omitted during the estimating process for the Box-Cox power transformation parameter. This enables analysts to transform variables earlier in the model building, making the Box-Cox transformation much simpler to apply in practice. They also noted that the Box–Cox transformation works better only if the cluster sizes are very large; and it is necessary to run a grid search of the transformation in order to determine the parameter estimate that maximizes the residual likelihood during the optimization process both under the linear and the mixed model settings. Nawata (1994) proposes a scanning ML method. Basically one conducts the entire methodology on a grid of fixed values of λ and then optimizes over this grid. Nawata (2013) used this method to calculate the ML estimator of the Box–Cox transformation model. Gurka et al. (2006) noted that

it is necessary to discuss how the estimation of λ affects inference about the other model parameters when one extends the Box–Cox transformation to the linear mixed model. It seems that it is possible for there to be a trade-off between transformation and mixed-effects models — both change the nature of the variance explained by the model.

Estimation of λ using profile maximum log–likelihood was discussed by Box and Cox (1964). The likelihood in relation to the original observations was obtained by multiplying the Jacobian of the transformation by the normal density. The ML estimates were found by taking the derivative of the log–likelihood function with respect to the regression parameters, setting the resulting derivatives equal to zero, solving the resulting equations, and replacing the results back into the log–likelihood function, and thus obtaining a profile log–likelihood function for λ. The value of λ that maximizes the profile log–likelihood was selected to be the best estimate of λ. A confidence interval based on the chi–squared χ^2 distribution was used to round the optimum λ to the nearest half. However, it is not possible to get a useful confidence interval in non–parametric situations because of the discrete nature of the underlying distributions. Hence, when faced with the decision on whether or not needing to transform the response, not only the best estimate of λ but also the relevant model selection criteria should be taken into account. Piepho and McCulloch (2004) considered the model selection in mixed models with transformations as "a difficult problem". Gurka (2004) suggested the use of the likelihood-based measures such as AIC and BIC to compare non-nested models. As already mentioned, the model with the lowest AIC or BIC is considered the best one. Furthermore, graphical measures can be used for exploring normality and homogeneity of variance such as

control charts, probability plots, histograms of residuals. Piepho and McCulloch (2004) suggested to fit a number of models and compare their fits by plotting the residual on the transformed and the untransformed scales.

1.3 Software review

The EM algorithm is an iterative method to find maximum likelihood estimators that may require a large number of iterations. This can be difficult and time consuming. Therefore, software is required, such as the open source statistics software, R. Aitkin's (1996a) NPML algorithm is implemented in R function `alldist()` in the **npmlreg** package, which is designed to account for simple overdispersion models using the NPML estimation, while variance component models (Aitkin, 1999a) can be fitted with `allvc()` in the **npmlreg** package (Aitkin et al., 2009; Einbeck et al., 2014). Einbeck and Hinde (2006) provided a guidance for performing NPML estimates for exponential families with unspecified mixing density.

The Box-Cox transformations are usually implemented in form of a plot of the profile log-likelihood for the univariate linear model against a set of λ values to locate the maximum, yielding transformed data that has constant variance and it follows a normal distribution more closely than the untransformed data. For an implementation of the Box-Cox transformations for the univariate linear model in R, see the `boxcox()` function in the **MASS** package (Venables and Ripley, 2002).

The R package **boxcoxmix** implements the methodology developed in this thesis in R (Almohaimeed and Einbeck, 2017), which is available from the Compre-

hensive R Archive Network (CRAN) at `https://cran.r-project.org/package=boxcoxmix`.

1.4 Summary

The thesis proposes a transformation approach by extending the Box–Cox transformation to overdispersion and two–level data scenarios in linear models as well as logistic mixed-effects models. In linear models, the aim is to ensure the constancy of error variance and the validity of a normal response distribution, whereas in mixed–effects binary regression models the Box-Cox transformation is used as an alternative link function for linearizing purposes. Using the Box–Cox power transformation in the presence of random effects that do not require any parametric assumptions on their distribution can be achieved by using the "Nonparametric Profile Maximum Likelihood" (NPPML) technique. To the best of my knowledge, the approach turns out to be the only one of its kind that has implemented the Box–Cox power transformation of the linear and logistic mixed effects models with unspecified random effect distributions. Simulated and real data are investigated using the R package **boxcoxmix** (Almohaimeed and Einbeck, 2017).

The remainder of the thesis is organized as follows. Chapter 2 begins by discussing the Box–Cox transformation for the linear model, as well as the theory and methodology underlying random effect models with unspecified random effect distribution. After that, it uses the NPPML technique to combine these two methods. In Chapter 3, we extend the Box–Cox transformation to the two–level structure using

the NPPML approach similarly as before, with some related adjustments. Chapter 4 provides a new way of implementing the work by Guerrero and Johnson (1982) that applied the Box–Cox transformation for the logistic regression model. Chapter 5 proposes an extension of the transformation to mixed-effects logistic models using the NPPML technique. Chapters 2, 3, 4 and 5 all follow the same basic format. They first present the theory and methodology underlying each model. The proposed approach will be then applied to that model. Finally, real and simulated data applications are used to verify the proposed approach. Chapter 6 concludes the thesis and gives an outlook to future work.

Chapter 2

Box-Cox transformations for random effect models

2.1 Introduction

Box and Cox (1964) introduced their transformation originally for the linear model, where it is assumed that a set of explanatory variables x_i, $i = 1, \ldots, n$, and a response variable y_i are linearly related such that $y_i = x_i^T \beta + \varepsilon_i$, with independent errors ε_i which are usually taken to be Gaussian and homoscedastic. The transformation $y_i^{(\lambda)}$ given in (1.1.1) is designed to mitigate violations of the latter two properties. However, not all types of violations can be mitigated through this route. It is often the case that the population from which the data are sampled consists of heterogeneous subpopulations. If these subpopulations are known, then they can simply be accounted for through an additional covariate in the model. However, frequently the subpopulations are *latent*, *i.e.* it is not possible to identify to which subpopulations

the observations of a sample belong (Wang, 2004). Under the resulting unobserved heteorogeneity, the errors cease to be independent, and their distribution tends to be multimodal. Fortunately, there is a well–known solution to this problem: The contribution by the latent subpopulation is captured by a random effect, conditional on which the errors restore their independence.

In this chapter, we intend to connect and combine both approaches, *i.e.* we assume that there is a value of λ so that the responses y_i are independently and normally distributed with mean function $E(y_i^{(\lambda)}|z_i) = x_i^T\beta + z_i$, conditionally on the random effect z_i. In explicit notation, one has then

$$y_i^{(\lambda)}|z_i \sim N(x_i^T\beta + z_i, \sigma^2), \tag{2.1.1}$$

where z_i is a random effect term with some density $g(\cdot)$. Note that under the presence of a random effect, the parametric intercept term can be omitted from $x_i^T\beta$. For the distribution of $g(\cdot)$, several choices are possible, among them the normal distribution, as in the classical literature on linear mixed models. The extension of the transformation under this scenario was proposed by Gurka et al. (2006), and extended to the longitudinal data setting by Maruo et al. (2017) whose main interests were in robust estimation of fixed (treatment) effects.

However, a normal distribution is by definition unimodal, and hence may fail to capture the full heterogeneity of the latent subpopulations. An obvious concern is whether there are any harmful effects of this potential misspecification. Agresti et al. (2004) showed that a misspecification of the random effects distribution may affect the prediction accuracy of the random effects as well as the fixed effects, and suggest

that "the safest approach might seem to be always to use a nonparametric rather than a parametric approach for the random effects distribution." In consideration of the random effects misspecification, Wang et al. (2012) argued that even when the estimation of the fixed effect is robust, the estimation of the random effects could be invalid.

Accordingly, we follow in this chapter the concepts laid out by Aitkin (1996a), which allows leaving the density $g(\cdot)$ unspecified. For estimation purposes, $g(\cdot)$ is then approximated by a finite discrete mixture with masses π_k at mass points z_k, $k = 1, \ldots, K$. These mixture parameters can be estimated alongside the other regression parameters in a usual EM algorithm. While it could, superficially, be argued that a 'discrete random effect' constitutes an even stronger limitation than a normal random effect, there is solid evidence that this is not the case. Methodologically, what is being approximated is the marginal likelihood,

$$L = \prod_{i=1}^{n} \int f(y_i|z_i)g(z_i)dz_i \approx \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f(y_i|z_k) \tag{2.1.2}$$

(where in our context $f(y_i|z_i)$ is the conditional density of the raw — not the transformed — data, which can be obtained from (2.1.1) using the transformation formula for probability density functions). It is known from early work by Laird (1978), Bock and Aitkin (1981) and Lindsay (1983), that this integral can be approximated with very high accuracy, and that the NPML estimate of the mixing distribution involves a finite number K of mass–points and corresponding masses. In practical applications, this integer K is typically very small, with values between $K = 2$ and 10.

In the context of model (2.1.1), the parameter λ needs to be estimated

on top of the regression and mixture parameters, which leads us to an approach which one can consider as a 'nonparametric profile maximum likelihood' (NPPML) technique, in a direct extension of the profile maximum log–likelihood estimation technique discussed by Box and Cox (1964).

The chapter is organised as follows. Section 2.2 presents the concept and computation of the Box–Cox transformation in univariate linear models (Box and Cox, 1964) along with a real data example. In Section 2.3, the random effect model is considered without making any specific assumptions about the mixing distribution of the random effects. The non–parametric maximum likelihood (NPML) approach advocated by Aitkin (1999a) is used to estimate the unspecified mixing distribution. For maximizing finite mixture likelihoods, we use the expectation–maximization (EM) algorithm that is iterated between adjusting given weights and using the current weights to calculate the parameter estimates. An extensive discussion of overdispersion in generalized linear models is to be found in Aitkin (1999a), Aitkin et al. (2005, 2009) and Einbeck et al. (2007). Einbeck and Hinde (2006) give a good practical introduction to the application of overdispersed generalized linear models using the software package **npmlreg** (Einbeck et al., 2014).

After discussing the Box–Cox transformation and the overdispersed generalized linear model, we consider combining these methods together in Section 2.4. We assume that the power transformation of the responses results in a linear model, following the same approach given by Box and Cox (1964) for transforming the response in univariate linear models. We extend this transformation to random effect models using the NPML technique, which, for our purposes, is adapted to-

wards a NPPML technique. Section 2.5 provides technical specifications including a non–iterative solution for the estimate of the model parameter, an introduction of a new approach to the standard error of the estimate of our model, a discussion of the choice of starting points and a brief description of the first cycle of the EM algorithm with NPPML estimation. Furthermore, in Section 2.6, we introduce a new R package **boxcoxmix** that implements the proposed approach in R (Almohaimeed and Einbeck, 2017), which is available from the Comprehensive R Archive Network (CRAN) at `https://cran.r-project.org/package=boxcoxmix`. Applications to simulated data sets in Section 2.7 and to real data sets in Section 2.9 demonstrate the accuracy and the efficiency of the proposed approach.

In Section 2.10, we consider the special case of the Box–Cox transformation for a random effect model without any independent variables, we call it a 'pure mixture model' to distinguish it from a more general type of mixture model 'mixed effect model'. We also illustrate the application of the approach to pure mixture models using real data examples. The Chapter concludes with a discussion in Section 2.12.

2.2 Box–Cox transformation

The Box–Cox transformation (Box and Cox, 1964) has been widely used in applied data analysis. The objective of the transformation is to select an appropriate parameter λ which is then used to induce constant variance and normality. It includes any positive or negative power, as well as the log. The transformation of the responses

y_i, takes the form:

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_i & (\lambda = 0) \end{cases} \tag{2.2.1}$$

where the restriction $y_i > 0$, $i = 1, \ldots, n$, applies. Note that when λ approaches zero,

$$\begin{aligned} \frac{y_i^{\lambda} - 1}{\lambda} = \frac{e^{\log y_i^{\lambda}} - 1}{\lambda} &= \frac{e^{\lambda \log y_i} - 1}{\lambda} \\ &\approx \frac{1 + \lambda \log y_i - 1}{\lambda} = \log y_i. \end{aligned}$$

Box and Cox (1964) also proposed a shifted power transformation to include zero and negative values of y_i, that is of the form

$$y_i^{(\lambda)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1 - 1}}{\lambda_1} & (\lambda_1 \neq 0), \\ \log(y_i + \lambda_2) & (\lambda_1 = 0) \end{cases} \tag{2.2.2}$$

where $\lambda = (\lambda_1, \lambda_2)$, λ_1 is the transformation parameter and λ_2 is chosen such that $y_i > -\lambda_2$. The discussion in this thesis is based on equation (2.2.1). So the focus here is on just positive responses.

2.2.1 Estimation of the model parameters

Let $\phi(y; \cdot, \cdot)$ denote the univariate Gaussian probability density function, with mean and variance specified in the remaining two function arguments. It is assumed that there is a value of λ for which the transformed observation $y^{(\lambda)}$ is independently normally distributed with parameters β and σ^2. Let $J(y, \lambda)$ be the Jacobian of the

transformation from y to $y^{(\lambda)}$, such that if $\lambda \neq 0$

$$J(y_i, \lambda) = \frac{\mathrm{d}y_i^{(\lambda)}}{\mathrm{d}y_i} = \frac{\lambda y_i^{\lambda-1} - 0}{\lambda} = y_i^{\lambda-1} \tag{2.2.3}$$

and if $\lambda = 0$

$$J(y_i, \lambda) = \frac{\mathrm{d}y_i^{(\lambda)}}{\mathrm{d}y_i} = \frac{1}{y_i} = y_i^{-1} \tag{2.2.4}$$

Setting $\lambda = 0$ in Equation (2.2.3) results in Equation (2.2.4), therefore, we will use Equation (2.2.3) in both cases.

The probability density of any single observation y is given by

$$f(y) = \phi(y^{(\lambda)}; \beta, \sigma^2) y^{\lambda-1} \tag{2.2.5}$$

where the last term is the Jacobian of the transformation multiplied by the normal density. Hence, the likelihood in relation to the original observations is

$$L(\lambda, \beta, \sigma^2) = \prod_{i=1}^{n} f(y_i) = \prod_{i=1}^{n} \frac{y_i^{\lambda-1}}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp\Big[-\frac{1}{2\sigma^2}(y_i^{(\lambda)} - x^T\beta)^2\Big] \tag{2.2.6}$$

The log–likelihood is then

$$\log L(\lambda, \beta, \sigma^2) = -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i^{(\lambda)} - x^T\beta)^2 + (\lambda-1)\sum_{i=1}^{n}\log y_i \tag{2.2.7}$$

For fixed λ, the maximum likelihood estimates can be found by taking the derivative of the log–likelihood with respect to the parameters β and σ^2, setting the resulting derivatives equal to zero, and solving the resulting equations. Now, taking the derivative of Equation (2.2.7) with respect to β yields

$$\frac{\partial \log L}{\partial \beta} = -\frac{1}{2\sigma^2}\sum_{i=1}^{n} 2\,(-x_i)(y_i^{(\lambda)} - x_i^T\beta) = 0$$

$$\sum_{i=1}^{n} x_i x_i^T \beta = \sum_{i=1}^{n} x_i y_i^{(\lambda)}$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = (\sum_{i=1}^{n} x_i x_i^T)^{-1} (\sum_{i=1}^{n} x_i y_i^{(\lambda)}) \tag{2.2.8}$$

Equation (2.2.8) in matrix notation is

$$\hat{\beta}^{(\lambda)} = \left(X^T X\right)^{-1} \left(X^T Y^{(\lambda)}\right) \tag{2.2.9}$$

where $Y^{(\lambda)}$ is an $n \times 1$ vector of observations $y_i^{(\lambda)}$, $i = 1, \ldots, n$, and X is an $n \times p$ matrix,

$$Y^{(\lambda)} = \begin{pmatrix} y_1^{(\lambda)} \\ \vdots \\ \vdots \\ y_n^{(\lambda)} \end{pmatrix} \quad \text{and} \quad X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}. \tag{2.2.10}$$

Note that, Equation (2.2.9) is just standard least-squares (MLE under normality) estimation for β using the transformed response $Y^{(\lambda)}$. Now, taking the derivative of Equation (2.2.7) with respect to σ, setting it equal to 0 and solving the resulting equation yields

$$\frac{\partial \log L}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n} (y_i^{(\lambda)} - x_i^T \beta)^2 = 0$$

$$n = \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i^{(\lambda)} - x_i^T \beta)^2$$

$$n\sigma^2 = \sum_{i=1}^{n} (y_i^{(\lambda)} - x_i^T \beta)^2$$

$$\Longrightarrow \hat{\sigma}^{2(\lambda)} = \frac{\sum_{i=1}^{n} (y_i^{(\lambda)} - x_i^T \beta)^2}{n} \tag{2.2.11}$$

Equation (2.2.11) is again as expected based on residual sums of squares (RSS), but note that it is the MLE and not the (more usual) unbiased estimate from ANOVA.

Replacing the results into Equation (2.2.7), the profile log–likelihood function for fixed λ is thus

$$\ell_P(\lambda) = \log L(\lambda, \hat{\beta}^{(\lambda)}, \hat{\sigma}^{2(\lambda)}) = -\frac{n}{2}\log 2\pi - n\log\hat{\sigma} - \frac{n\hat{\sigma}^{2(\lambda)}}{2\hat{\sigma}^{2(\lambda)}} + (\lambda-1)\sum_{i=1}^{n}\log y_i$$
$$= -\frac{n}{2}\log 2\pi - n\log\hat{\sigma}^{(\lambda)} - \frac{n}{2} + (\lambda-1)\sum_{i=1}^{n}\log y_i \quad (2.2.12)$$

The profile maximum log-likelihood estimate of λ is thus

$$\hat{\lambda} = arg\ \max_{\lambda}\ \ell_P(\lambda). \quad (2.2.13)$$

For this, one needs to define the range over which the optimization of λ will occur and this range must include 1. For each λ, the MLE of β and σ^2 is computed using $\hat{\beta} = \hat{\beta}^{(\lambda)}$ and $\hat{\sigma}^2 = \hat{\sigma}^{2(\lambda)}$ and then $\ell_P(\hat{\lambda})$ is maximized over a given grid of values for λ using the values for $\hat{\beta}^{(\lambda)}$ and $\hat{\sigma}^{2(\lambda)}$ given in Equations (2.2.9) and (2.2.11).

An approximate $100(1-\alpha)\%$ likelihood–ratio based confidence interval for λ can be obtained by

$$\ell_P(\hat{\lambda}) - \ell_P(\lambda) < \frac{1}{2}\chi^2_{\alpha,1} \quad (2.2.14)$$

where $\chi^2_{\alpha,1}$ is the value of the chi–square statistic with 1 degree of freedom.

2.2.2 Existing R implementation: `boxcox()`

The Box–Cox approach for the linear model is implemented in the **MASS** function `boxcox()` in R (Venables and Ripley, 2002). The best estimator of λ is selected according to Equation (2.2.13) via the function `boxcox()` which plots the profile log–likelihood for a range of λ values, including a vertical line indicating the maximum value of λ. It uses a 95% confidence limit to define the range of the optimum λ, in

which the optimal choice for λ can occur anywhere within the confidence limits. A λ value of 1 does not change the shape of the distribution, therefore, a confidence interval that includes the value 1, corresponds to no transformation.

Example 2.2.1. the Pennsylvanian Hospital Stay data

The Pennsylvanian Hospital Stay (`hosp`) dataset, that is part of the R package **npmlreg** (Einbeck et al., 2014) and consists of 25 observations, is used with the function `boxcox()` in R to produce a plot of the profile likelihood function which summarises information concerning λ, including a horizontal line indicating the critical value of the likelihood ratio at the 95% confidence level (see Figures 2.2.1(a) and 2.2.2(a)). A normal probability plot (QQ–plot) can be then used to assess the fit of the data before and after the transformation to a normal distribution (see Figures 2.2.1(b) and 2.2.2(b)). If the data fits a normal distribution, the points in the QQ–plot lie along a straight diagonal line. To investigate the effects of the covariates `age`, `sex` and `temp1` on the total number of days patients spent in hospital (`duration`), where `age` denotes the age of patient in whole years, `sex` denotes the gender (`1=Male, 2=Female`) and `temp1` denotes the first measured temperature following admission, measured in Fahrenheit, the following model is fitted to the `hosp` data,

$$y_i^{(\lambda)} = \beta_0 + \beta_1 \cdot \texttt{age}_i + \beta_2 \cdot \texttt{sex}_i + \beta_3 \cdot \texttt{temp1}_i + \varepsilon_i \tag{2.2.15}$$

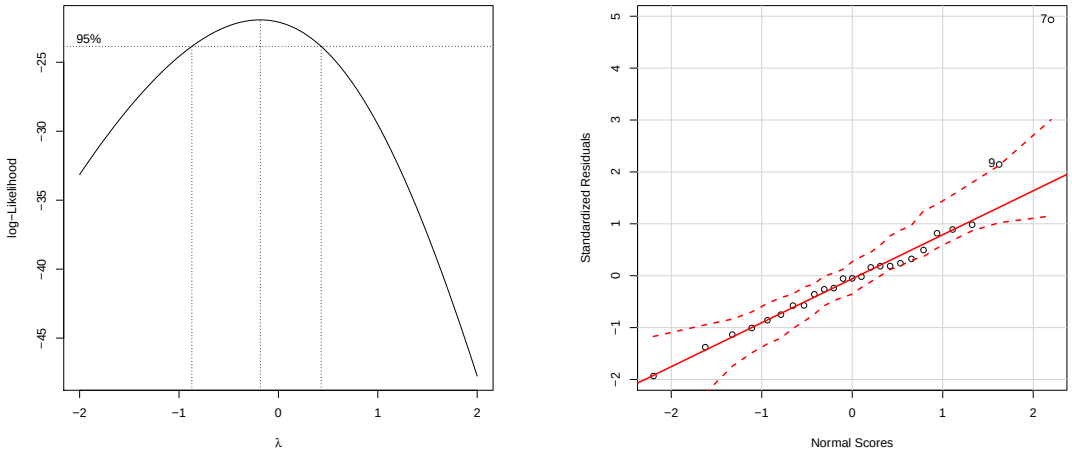
The code below produces the Box–Cox transformation plot.

R Note:

Import the `hosp` data into R, then:

```
library(MASS)
```

```
boxcox(duration ~ age + sex + temp1, data=hosp)
```



(a) Profile log–likelihood plot for λ of the untransformed data

(b) Probability plot of the untransformed data

Figure 2.2.1: Untransformed `hosp` data

The Box–Cox plot is shown in Figure 2.2.1(a). The 95% confidence interval does not include the value one, indicating that the data support the need for the transformation. Since the value of $\hat{\lambda} = -0.2$ is close to zero, the natural log transformation would be appropriate. The following code applies a logarithmic transformation of the hosp data,

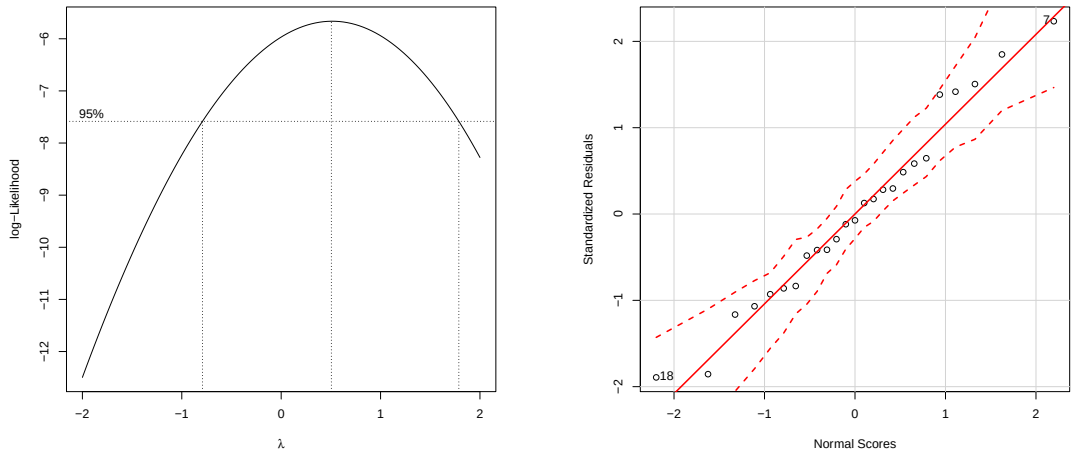
R Note:

```
boxcox(log(duration) ~ age + sex + temp1, data=hosp)
```

Here the response is a duration, so like a time to event in which case $Var(Y) \approx \mu^2$ and a log-transformation would be variance-stabilising

$$Var(\log(Y)) \approx \frac{1}{\mu^2} Var(Y) \approx \text{constant}$$

this is a special case of Equation (1.1.3) with $\lambda = 0$ and $\delta = 0$.



(a) Profile log–likelihood plot for λ of the transformed data

(b) Probability plot of transformed data

Figure 2.2.2: Transformed `hosp` data

After the transformation, we can see that the 95% confidence interval in Figure 2.2.2(a) contains 1. This gives further support to the decision to use the natural log transformation. Additionally, by comparing the probability plot of the residuals from the univariate model of the transformed response ($\hat{\varepsilon}_i^{(\lambda)} = y_i^{(\lambda)} - \hat{y}_i{}^{(\lambda)} = y_i^{(\lambda)} - x_i^T \hat{\beta}^{(\lambda)}$) in Figure 2.2.2(b) with that for the untransformed model ($\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - x_i^T \hat{\beta}$) in Figure 2.2.1(b), it is obvious that the residuals of the untransformed data have two clear outliers while the residuals of the transformed data match a normal distribution better.

2.3 Random effects

We consider the linear model in which an unobserved random effect z_i with an unspecified distribution $g(z)$ is added to the linear predictor $x_i^T\beta$ for the i-th observation. The responses y_i are independently distributed with mean function $E(y_i|z_i) = x_i^T\beta + z_i$ and variance function $Var(y_i|z_i) = \sigma^2$, conditionally on the random effect z_i. The marginal mean is

$$\begin{aligned} E(y_i) &= E(E(y_i|z_i)) = \\ &= \int E(y_i|z_i)\, g(z_i) dz_i \\ &= \int (x_i^T\beta + z_i)\, g(z_i) dz_i \\ &= \int x_i^T\beta\, g(z_i) dz_i + \int z_i\, g(z_i) dz_i \\ &= x_i^T\beta + E(z_i) \end{aligned} \tag{2.3.1}$$

where the term $x_i^T\beta$ does not include an intercept, hence $E(z_i)$ can be thought of as an intercept β_0. The marginal variance is

$$\begin{aligned} Var(y_i) &= E(Var(y_i|z_i)) + Var(E(y_i|z_i)) = \\ &= E(Var(y_i|z_i)) + Var(x_i^T\beta + z_i) = \\ &= E(\sigma^2) + Var(z_i) \\ &= \sigma^2 + Var(z_i) \end{aligned} \tag{2.3.2}$$

In the case of a normal random effect $z_i \sim N(0, \tau^2)$, this would imply $y_i \sim N(x_i^T\beta, \tau^2 + \sigma^2)$. The conditional probability density function of y_i given z_i is

given by

$$f(y_i|z_i) = \phi(y_i; x_i^T\beta + z_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_i - x_i^T\beta - z_i)^2\right]. \qquad (2.3.3)$$

2.3.1 Estimation of finite mixtures

Parameter estimation requires maximizing the likelihood

$$L(\beta, \sigma^2, g) = \prod_{i=1}^{n} \int f(y_i|z_i)g(z_i)dz_i \qquad (2.3.4)$$

Under the non-parametric maximum likelihood (NPML) approach, the integral over the (unspecified) mixing distribution $g(z)$ is approximated by a discrete distribution on a finite number K of mass–points z_k, with masses π_k (Aitkin et al., 2009). The K components of the finite mixture satisfy the relations $\sum_{k=1}^{K} \pi_k = 1$; $0 < \pi_k \leqslant 1$. The approximated likelihood is then

$$L(\beta, \sigma^2, z_1, ..., z_k, \pi_1, ..., \pi_k) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_{ik} \qquad (2.3.5)$$

where $f_{ik} = f(y_i|z_k)$. The log-likelihood is then

$$\ell = \log L = \log\left(\prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_{ik}\right) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k f_{ik}\right) \qquad (2.3.6)$$

Since Equation (2.3.6) is intractable we augment the data structure by defining indicators G_{ik} such that

$$G_{ik} = \begin{cases} 1 & \text{if case } i \text{ stems from component } k, \\ 0 & \text{otherwise.} \end{cases} \qquad (2.3.7)$$

Now the "complete data" likelihood would be

$$L^* = \prod_{i=1}^{n} \prod_{k=1}^{K} (\pi_k f_{ik})^{G_{ik}}. \tag{2.3.8}$$

The complete log-likelihood is thus

$$\begin{aligned} \ell^* = \log L^* &= \log \Big(\prod_{i=1}^{n} \prod_{k=1}^{K} (\pi_k f_{ik})^{G_{ik}} \Big) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} \log \Big((\pi_k f_{ik})^{G_{ik}} \Big) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} G_{ik} \log \Big(\pi_k f_{ik} \Big) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} \Big[G_{ik} \log \pi_k + G_{ik} \log f_{ik} \Big] \end{aligned} \tag{2.3.9}$$

where

$$\begin{aligned} \log f_{ik} &= \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2\sigma^2} (y_i - x_i^T \beta - z_k)^2 \right] \right) \\ &= \left(-\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (y_i - x_i^T \beta - z_k)^2 \right), \end{aligned} \tag{2.3.10}$$

then

$$\ell^* = \sum_{i=1}^{n} \sum_{k=1}^{K} \left[G_{ik} \log \pi_k + G_{ik} \left(-\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (y_i - x_i^T \beta - z_k)^2 \right) \right]. \tag{2.3.11}$$

We now apply the expectation-maximization (EM) approach to find the maximum likelihood estimate (MLE) of the model parameters. Given some starting values β^0, σ^0, z_k^0, and π_k^0 (the choice of which will be discussed below), set $\hat{\beta} = \beta^0$, $\hat{\sigma} = \sigma^0$, $\hat{z}_k = z_k^0$, $\hat{\pi}_k = \pi_k^0$, $k = 1, 2, \ldots, K$, and iterate between the following steps:

Expectation step (E-step): As G_{ik} are unknown, we use the conditional expectation of the log-likelihood to replace G_{ik},

$$
\begin{aligned}
E[G_{ik}|y_i] &= P(G_{ik}=1|y_i) = \frac{P(G_{ik}=1)P(y_i|G_{ik}=1)}{P(y_i)} \\
&= \frac{\pi_k \exp\left[-\frac{1}{2\sigma^2}(y_i - x_i^T\beta - z_k)^2\right]}{\sum_\ell \pi_\ell \exp\left[-\frac{1}{2\sigma^2}(y_i - x_i^T\beta - z_\ell)^2\right]} = \frac{\pi_k f_{ik}}{\sum_\ell \pi_\ell f_{i\ell}} \equiv w_{ik}
\end{aligned} \tag{2.3.12}
$$

where w_{ik} is the posterior probability that observation y_i comes from component k, and f_{ik} depends via equation (2.3.3) implicitly on the current values of $\hat{z}_k$, $\hat{\beta}$ and $\hat{\sigma}^2$.

Maximization step (M-step): Calculate $\hat{z}_k, \hat{\sigma}^2, \hat{\beta}$ and $\hat{\pi}_k$ using current w_{ik},

$$
\begin{aligned}
\frac{\partial \ell^*}{\partial z_k} = -\frac{1}{2\sigma^2}(2)\sum_{i=1}^n w_{ik}(y_i - x_i^T\beta - z_k)(-1) &= 0 \\
\sum_{i=1}^n w_{ik}(y_i - x_i^T\beta - z_k) &= 0 \\
\sum_{i=1}^n w_{ik}y_i - \sum_{i=1}^n w_{ik}x_i^T\beta - \sum_{i=1}^n w_{ik}z_k &= 0 \\
\sum_{i=1}^n w_{ik}z_k = \sum_{i=1}^n w_{ik}y_i - \sum_{i=1}^n w_{ik}x_i^T\beta & \\
\Longrightarrow \hat{z}_k = \frac{\sum_{i=1}^n w_{ik}(y_i - x_i^T\beta)}{\sum_{i=1}^n w_{ik}} &
\end{aligned} \tag{2.3.13}
$$

Similarly

$$
\begin{aligned}
\frac{\partial \ell^*}{\partial \beta} = -\frac{1}{2\sigma^2}(2)\sum_{i=1}^n\sum_{k=1}^K w_{ik}(-x_i)(y_i - x_i^T\beta - z_k) &= 0 \\
\sum_{i=1}^n\sum_{k=1}^K w_{ik}x_i(y_i - x_i^T\beta - z_k) &= 0 \\
\sum_{i=1}^n\sum_{k=1}^K w_{ik}x_iy_i - \sum_{i=1}^n\sum_{k=1}^K w_{ik}x_ix_i^T\beta - \sum_{i=1}^n\sum_{k=1}^K w_{ik}x_iz_k &= 0 \\
\sum_{i=1}^n x_iy_i\sum_{k=1}^K w_{ik} - \sum_{i=1}^n x_ix_i^T\beta\sum_{k=1}^K w_{ik} - \sum_{i=1}^n x_i\sum_{k=1}^K w_{ik}z_k &= 0 \\
\sum_{i=1}^n x_iy_i - \sum_{i=1}^n x_ix_i^T\beta - \sum_{i=1}^n x_i\sum_{k=1}^K w_{ik}z_k &= 0 \\
\Longrightarrow \hat{\beta} = \left(\sum_{i=1}^n x_ix_i^T\right)^{-1}\left(\sum_{i=1}^n x_iy_i - \sum_{i=1}^n x_i\sum_{k=1}^K w_{ik}z_k\right) &
\end{aligned}
$$

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i x_i^T \right)^{-1} \sum_{i=1}^{n} x_i \left(y_i - \sum_{k=1}^{K} w_{ik} z_k \right) \tag{2.3.14}$$

Equation (2.3.14) in matrix notation is

$$\hat{\beta} = \left(X^T X \right)^{-1} X^T (Y - WZ) \tag{2.3.15}$$

where Y is an $n \times 1$ vector of observations y_i, $i = 1, \ldots, n$, and X is an $n \times p$ matrix,

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}, \qquad X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}, \tag{2.3.16}$$

W is an $n \times K$ matrix and Z is a $K \times 1$ vector,

$$W = \begin{pmatrix} w_{11} & \cdots & \cdots & w_{1K} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & \cdots & \cdots & w_{nK} \end{pmatrix} \quad \text{and} \quad Z = \begin{pmatrix} z_1 \\ \vdots \\ \vdots \\ z_K \end{pmatrix} \tag{2.3.17}$$

$\hat{z}_k$ and $\hat{\beta}$ are obtained by iterating between Equations (2.3.13) and (2.3.14) a small number of times in each M-step. The MLE for σ is

$$\begin{aligned} \frac{\partial \ell^*}{\partial \sigma} = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} \left[-\frac{1}{\sigma} + \frac{1}{\sigma^3} (y_i - x_i^T \beta - z_k)^2 \right] = 0 \\ -\sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} + \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{w_{ik}}{\sigma^2} (y_i - x_i^T \beta - z_k)^2 = 0 \\ n = \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{w_{ik}}{\sigma^2} (y_i - x_i^T \beta - z_k)^2 \\ n\sigma^2 = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik} (y_i - x_i^T \beta - z_k)^2 \end{aligned}$$

$$\Longrightarrow \hat{\sigma}^2 = \sum_{i=1}^{n}\sum_{k=1}^{K} \frac{w_{ik}(y_i - x_i^T\beta - z_k)^2}{n} \tag{2.3.18}$$

Since $\sum_{k=1}^{K} \pi_k = 1$, we apply a lagrange multiplier,

$$\begin{aligned}
&\frac{\partial\left(\ell^* - \theta(\sum_{k=1}^{K}\pi_k - 1)\right)}{\partial \pi_k} = 0 \qquad k = 1, ..., K \\
&\Longrightarrow \frac{\sum_{i=1}^{n} w_{ik}}{\pi_k} - \theta = 0 \quad \Longrightarrow \hat{\pi}_k = \frac{\sum_{i=1}^{n} w_{ik}}{\theta} \\
1 = \sum_{k=1}^{K}\pi_k = \sum_{k=1}^{K}\sum_{i=1}^{n}\frac{w_{ik}}{\theta} &= \frac{1}{\theta}\sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} = \frac{1}{\theta}\cdot n \Rightarrow \theta = n \\
&\Longrightarrow \hat{\pi}_k = \frac{\sum_{i=1}^{n} w_{ik}}{n}
\end{aligned} \tag{2.3.19}$$

where $\hat{\pi}_k$ is the average posterior probability for component k.

2.3.2 Existing R implementation: `alldist()`

One can use the **npmlreg** (Einbeck et al., 2014) function `alldist()` to fit random effect models. However, the function `alldist()` relies on the output of the function `glm()` rather than computing (2.3.13), (2.3.14), (2.3.18) and (2.3.19) directly. For starting values of the EM algorithm one can use Gauss-Hermite quadrature points (Einbeck and Hinde, 2006):

$$z_k^0 = \hat{\beta}_0 + \texttt{tol} \times s \times g_k \tag{2.3.20}$$

where β_0 is the intercept of the fitted model $y_i = x_i^T\beta + \varepsilon_i$, `tol` is a scaling parameter restricted to the choice $0 \leq \texttt{tol} \leq 2$, g_k are Gauss-Hermite quadrature points with masses π_k^0, and s is the standard deviation of residuals defined as,

$$s = \sqrt{\frac{1}{n-p}\sum_{i=1}^{n}\hat{\varepsilon}_i^{\,2}} \tag{2.3.21}$$

where $n - p$ is the degrees of freedom for the residuals $\hat{\varepsilon}_i$, n is the sample size, p represents the number of parameters used to fit the model and the residual is the difference between the observed data of the dependent variable y_i and the fitted values $\hat{y}_i$ (i.e. $\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - x_i^T \hat{\beta}$).

Example 2.3.1. the Strength data

We consider the `strength` data from the R library **mdscore** (da Silva-Júnior et al., 2014) which is a subsample of the 5 x 2 factorial experiment of 30 observations given by Ostle and Malone (1954). The objective here is to investigate the effects of the covariates `lot` and `cut` on the impact strength, where `lot` denotes the lot of the material (I, II, III, IV, V) and `cut` denotes the type of specimen cut (Lengthwise, Crosswise). The random effect model that is fitted to the `strength` data is a two-way `lot` $\times$ `cut` interaction model. For the i-th `cut` and j-th `lot`, we have

$$\eta_{ij} = \gamma_i + \beta_j + \delta_{ij} + z, \qquad i = 1, 2, \quad j = 1, 2, .., 5, \tag{2.3.22}$$

where $\gamma_1 = 0$, $\beta_1 = 0$, $\delta_{1,1} = \delta_{1,2} = \cdots = \delta_{1,5} = \delta_{2,1} = 0$, and z is the random effect with an unspecified mixing distribution. Under the NPML approach, $g(z)$ is approximated by a discrete distribution on a finite number K of mass–points z_k, with masses π_k.

R Note:

Import the `strength` data into R, then:

```
library(npmlreg)
```

```
fit<-alldist(y ~ cut*lot, data=strength, k=3)

summary(fit)

# Call:  alldist(formula = y ~ cut * lot, data = strength, k = 3)

#

# Coefficients:

#                         Estimate  Std. Error    t value

# cut Crosswise         -0.2493290 0.012165308 -20.495080

# lot II                -0.0796219 0.011737237  -6.783700

# lot III               -0.2659958 0.012165303 -21.865116

# lot IV                -0.2192806 0.012500052 -17.542378

# lot V                 -0.4996067 0.013048088 -38.289650

# cut Crosswise:lot II   0.3259945 0.016721037  19.496067

# cut Crosswise:lot III  0.1493025 0.016721344   8.928860

# cut Crosswise:lot IV   0.2660311 0.016720593  15.910387

# cut Crosswise:lot V    0.1730249 0.017367740   9.962431

# MASS1                  0.8934177 0.008602925 103.850457

# MASS2                  1.0422844 0.010236461 101.820778

# MASS3                  1.1631634 0.009931993 117.112784

#

# Mixture proportions:

#   MASS1      MASS2      MASS3

# 0.2333320  0.5660864  0.2005815

#
```

```
# Component distribution - MLE of sigma:    0.02265

# Random effect distribution - standard deviation:    0.08939663

#

# -2 log L:     -83.3     Convergence at iteration  12

plot(fit)
```

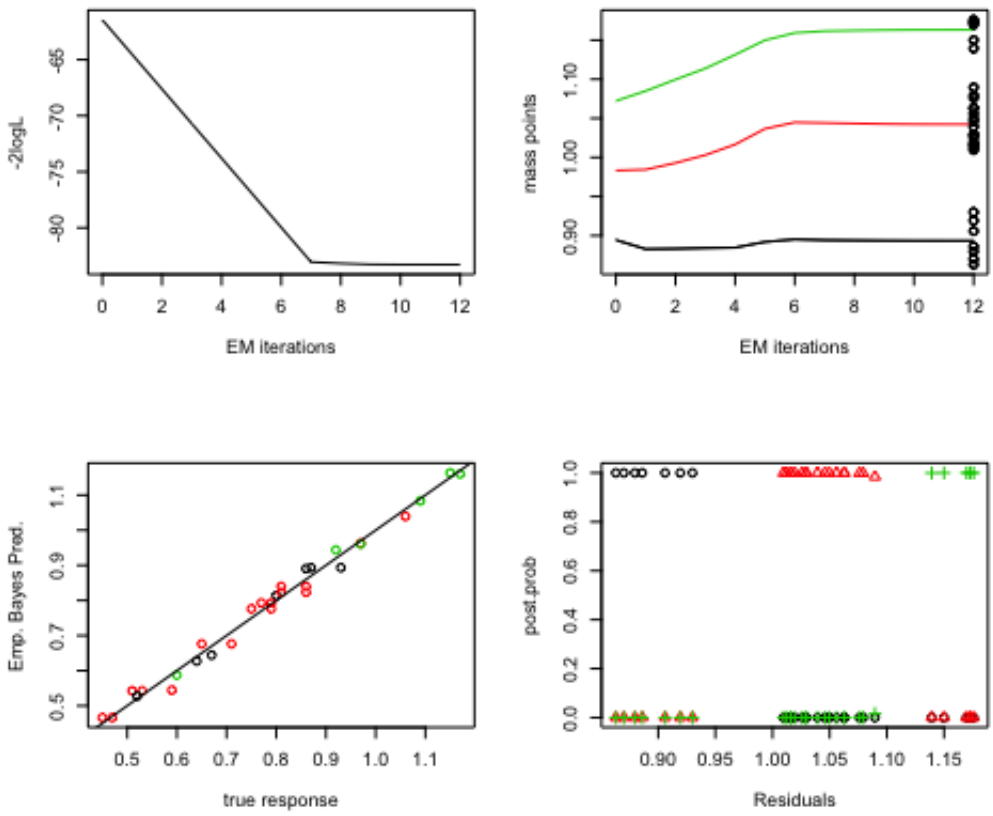


Figure 2.3.1: fitting the random effect with NPML to the `strength` data using the function `alldist()`, with `k=3` and `tol=0.5`

It is clear from the top right plot of Figure 2.3.1 that the three estimated mass points that are coloured by black (MASS1), red (MASS2) and green (MASS3) for $k = 1, 2$ and 3, respectively, are distinct and identifiable, suggesting that the random effects should be taken into account. In the bottom right plot of Figure 2.3.1, the residual on the x-axis is $\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - x_i^T\hat{\beta} - \hat{z}_i$, where $\hat{z}_i = \sum_{k=1}^{K} w_{ik}\hat{z}_k$ (Aitkin,

1996b), and the posterior probability on the y-axis corresponds to the weights w_{ik} in the final iteration of the EM algorithm that is the probability that the case i comes from component k and that can take on any values between 0 and 1. The posterior probability that lower residuals came from component 1 is 1 while that from components 2 and 3 is 0. Also, the posterior probability that middle residuals came from component 2 is 1 and that from components 1 and 3 is 0. Similarly, the posterior probability that upper residuals came from component 3 is 1, however, that from components 1 and 2 is 0.

2.4 Box-Cox transformations for random effect models

In this section, the Box-Cox transformation is extended to the random effects model. Recall the equation for the Box-Cox transformation of the response y_i above

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_i & (\lambda = 0) \end{cases} \tag{2.4.1}$$

and that for $y_i > 0$, $i = 1, ..., n$. From the inversion of (2.4.1) we get

$$\hat{y}_i = \begin{cases} \left(1 + \lambda \eta_i\right)^{1/\lambda} & (\lambda \neq 0), \\ e^{\eta_i} & (\lambda = 0) \end{cases} \tag{2.4.2}$$

where $\eta_i = x_i^T \beta + z_i$.

2.4.1 Estimation of finite mixtures

In the case of random effect models, it is assumed that there is a value of λ for which,

$$y_i^{(\lambda)}|z_i \sim N(x_i^T\beta + z_i, \sigma^2) \tag{2.4.3}$$

where z_i is a random effect with an unspecified $g(z_i)$ distribution. Taking account of the Jacobian of the transformation from y to $y^{(\lambda)}$, the conditional probability density function of y_i given z_i is

$$f(y_i, \lambda|z_i) = \frac{y_i^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_i)^2\right] \tag{2.4.4}$$

The likelihood can now be approximated using the NPML approach (Aitkin et al., 2009) as

$$L(\lambda, \beta, \sigma^2, g) = \prod_{i=1}^{n} \int f(y_i, \lambda|z_i)g(z_i)dz_i \approx \prod_{i=1}^{n}\sum_{k=1}^{K} \pi_k f_{ik}^{(\lambda)} \tag{2.4.5}$$

where $f_{ik}^{(\lambda)} = f(y_i, \lambda|z_k)$. The log-likelihood is then

$$\ell = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k f_{ik}^{(\lambda)}\right) \tag{2.4.6}$$

Using notation as defined in (2.3.7), the "complete data" likelihood would be

$$L^* = \prod_{i=1}^{n}\prod_{k=1}^{K} (\pi_k f_{ik}^{(\lambda)})^{G_{ik}}. \tag{2.4.7}$$

Now the complete log-likelihood would be

$$\ell^* = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[G_{ik}\log\pi_k + G_{ik}\log f_{ik}^{(\lambda)}\right]$$

where

$$\log f_{ik}^{(\lambda)} = \log\left(\frac{y_i^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_k)^2\right]\right)$$

$$= \left(-\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_k)^2 + (\lambda-1)\log y_i \right), \quad (2.4.8)$$

then

$$\ell^* = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[G_{ik}\log\pi_k + G_{ik}\left(-\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_k)^2 \right.\right.$$
$$\left.\left. +(\lambda-1)\log y_i \right)\right]. \quad (2.4.9)$$

If $K = 1$, the log-likelihood would be the usual log-likelihood of the Box–Cox model without random effects.

The EM algorithm can then be applied to find the MLE of the model parameters. As mentioned in the previous section, we could specify a set of parameters β^0, σ^0, z_k^0, and π_k^0, and start the iterative procedure:

E-step: As G_{ik} are unknown, we use the conditional expectation of the log-likelihood to replace G_{ik},

$$E[G_{ik}|y_i] = P(G_{ik}=1|y_i) = \frac{P(G_{ik}=1)P(y_i|G_{ik}=1)}{P(y_i)}$$
$$= \frac{\pi_k \exp\left[-\frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_k)^2\right]}{\sum_\ell \pi_\ell \exp\left[-\frac{1}{2\sigma^2}(y_i^{(\lambda)} - x_i^T\beta - z_\ell)^2\right]} = \frac{\pi_k f_{ik}^{(\lambda)}}{\sum_\ell \pi_\ell f_{i\ell}^{(\lambda)}} \equiv w_{ik}^{(\lambda)} \quad (2.4.10)$$

where $w_{ik}^{(\lambda)}$ is the posterior probability that observation $y_i^{(\lambda)}$ comes from component k, and $f_{ik}^{(\lambda)}$ depends via equation (2.4.4) implicitly on the current values of $\hat{z}_k$, $\hat{\beta}$ and $\hat{\sigma}^2$. Note that the Jacobian term cancels out as it does not depend on k.

M-step: Calculate $\hat{z}_k^{(\lambda)}, \hat{\sigma}^{2(\lambda)}, \hat{\beta}^{(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ using current $w_{ik}^{(\lambda)}$,

$$\frac{\partial \ell^*}{\partial z_k} = -\frac{1}{2\sigma^2}(2)\sum_{i=1}^{n} w_{ik}^{(\lambda)}(y_i^{(\lambda)} - x_i^T\beta - z_k)(-1) = 0$$
$$\sum_{i=1}^{n} w_{ik}^{(\lambda)}(y_i^{(\lambda)} - x_i^T\beta - z_k) = 0$$

$$\sum_{i=1}^{n} w_{ik}^{(\lambda)} y_i^{(\lambda)} - \sum_{i=1}^{n} w_{ik}^{(\lambda)} x_i^T \beta - \sum_{i=1}^{n} w_{ik}^{(\lambda)} z_k = 0$$

$$\sum_{i=1}^{n} w_{ik}^{(\lambda)} z_k = \sum_{i=1}^{n} w_{ik}^{(\lambda)} y_i^{(\lambda)} - \sum_{i=1}^{n} w_{ik}^{(\lambda)} x_i^T \beta$$

$$\implies \hat{z}_k^{(\lambda)} = \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)} (y_i^{(\lambda)} - x_i^T \beta)}{\sum_{i=1}^{n} w_{ik}^{(\lambda)}} \tag{2.4.11}$$

Similarly

$$\frac{\partial \ell^*}{\partial \beta} = -\frac{1}{2\sigma^2}(2) \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} (-x_i)(y_i^{(\lambda)} - x_i^T \beta - z_k) = 0$$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} x_i (y_i^{(\lambda)} - x_i^T \beta - z_k) = 0$$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} x_i y_i^{(\lambda)} - \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} x_i x_i^T \beta - \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} x_i z_k = 0$$

$$\sum_{i=1}^{n} x_i y_i^{(\lambda)} \sum_{k=1}^{K} w_{ik}^{(\lambda)} - \sum_{i=1}^{n} x_i x_i^T \beta \sum_{k=1}^{K} w_{ik}^{(\lambda)} - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k = 0$$

$$\sum_{i=1}^{n} x_i y_i^{(\lambda)} - \sum_{i=1}^{n} x_i x_i^T \beta - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k = 0$$

$$\implies \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i x_i^T \right)^{-1} \left(\sum_{i=1}^{n} x_i y_i^{(\lambda)} - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k \right)$$

$$= \left(\sum_{i=1}^{n} x_i x_i^T \right)^{-1} \sum_{i=1}^{n} x_i \left(y_i^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k \right) \tag{2.4.12}$$

and

$$\frac{\partial \ell^*}{\partial \sigma} = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left[-\frac{1}{\sigma} + \frac{1}{\sigma^3} (y_i^{(\lambda)} - x^T \beta - z_k)^2 \right] = 0$$

$$-\sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} + \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{w_{ik}^{(\lambda)}}{\sigma^2} (y_i^{(\lambda)} - xi^T \beta - z_k)^2 = 0$$

$$n = \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{w_{ik}^{(\lambda)}}{\sigma^2} (y_i^{(\lambda)} - xi^T \beta - z_k)^2$$

$$n\sigma^2 = \sum_{i=1}^{n} \sum_{k=1}^{K} w_{ik}^{(\lambda)} (y_i^{(\lambda)} - xi^T \beta - z_k)^2$$

$$\implies \hat{\sigma}^{2(\lambda)} = \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{w_{ik}^{(\lambda)} (y_i^{(\lambda)} - x^T \beta - z_k)^2}{n} \tag{2.4.13}$$

Since $\sum_{k=1}^{K} \pi_k = 1$, we apply a lagrange multiplier,

$$\begin{aligned}
&\frac{\partial\left(\ell^* - \theta(\sum_{k=1}^{K}\pi_k - 1)\right)}{\partial \pi_k} = 0 \qquad k = 1, ..., K \\
&\Longrightarrow \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)}}{\pi_k} - \theta = 0 \quad \Longrightarrow \pi_k = \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)}}{\theta} \\
1 = \sum_{k=1}^{K}\pi_k = \sum_{k=1}^{K}\sum_{i=1}^{n}\frac{w_{ik}^{(\lambda)}}{\theta} &= \frac{1}{\theta}\sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)} = \frac{1}{\theta}\cdot n \Rightarrow \theta = n \\
&\Longrightarrow \hat{\pi}_k^{(\lambda)} = \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)}}{n} \qquad (2.4.14)
\end{aligned}$$

where $\hat{\pi}_k^{(\lambda)}$ is the average posterior probability for component k. This leads to the four reconciled equations (the notation emphasizes the dependence on λ explicitly)

$$\hat{z}_k^{(\lambda)} = \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)}(y_i^{(\lambda)} - x_i^T\hat{\beta}^{(\lambda)})}{\sum_{i=1}^{n} w_{ik}} \qquad (2.4.15)$$

$$\hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1}\sum_{i=1}^{n} x_i\left(y_i^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)}\hat{z}_k^{(\lambda)}\right) \qquad (2.4.16)$$

$$\hat{\sigma}^{2(\lambda)} = \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{w_{ik}^{(\lambda)}(y_i^{(\lambda)} - x^T\hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)})^2}{n} \qquad (2.4.17)$$

$$\hat{\pi}_k^{(\lambda)} = \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)}}{n} \qquad (2.4.18)$$

Apparently, finding the MLE of the model parameters can be straightforward.

Equation (2.4.16) in matrix notation is

$$\hat{\beta}^{(\lambda)} = \left(\underbrace{\underbrace{X^T}_{\mathrm{p}\times\mathrm{n}}\underbrace{X}_{\mathrm{n}\times\mathrm{p}}}_{\mathrm{p}\times\mathrm{p}}\right)^{-1}\underbrace{X^T}_{\mathrm{p}\times\mathrm{n}}\left(\underbrace{\underbrace{Y^{(\lambda)}}_{\mathrm{n}\times 1} - \underbrace{\underbrace{W^{(\lambda)}}_{\mathrm{n}\times\mathrm{K}}\underbrace{\hat{Z}^{(\lambda)}}_{\mathrm{K}\times 1}}_{\mathrm{n}\times 1}}_{\mathrm{n}\times 1}\right) \qquad (2.4.19)$$

where $Y^{(\lambda)}$ is as in Equation (2.3.16). $\hat{z}_k^{(\lambda)}$ and $\hat{\beta}^{(\lambda)}$ are obtained by iterating between Equations (2.4.15) and (2.4.19) a small number of times in each M-step. Replacing the results into Equation (2.4.6) we get the non-parametric profile log-likelihood function.

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \left(-\frac{1}{2}\log 2\pi - \log \hat{\sigma}^{(\lambda)} - \frac{1}{2\hat{\sigma}^{2(\lambda)}} (y_i^{(\lambda)} - x_i^T \hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)})^2 + (\lambda - 1)\log y_i \right) \right] \quad (2.4.20)$$

Now let

$$\begin{aligned} \xi_{ik}^{(\lambda)} &= y_i^{(\lambda)} - x_i^T \hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)} && (2.4.21) \\ &= y_i^{(\lambda)} - x_i^T \hat{\beta}^{(\lambda)} - \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \\ &= \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} y_i^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_i^T \hat{\beta}^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \hat{\beta}^{(\lambda)}}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \\ \implies \xi_{ik}^{(\lambda)} &= \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} \left((y_i^{(\lambda)} - y_m^{(\lambda)}) - (x_i^T - x_m^T)\hat{\beta}^{(\lambda)} \right)}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} && (2.4.22) \end{aligned}$$

The non-parametric profile log-likelihood function is thus

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \left(-\frac{1}{2}\log 2\pi - \log \hat{\sigma}^{(\lambda)} - \frac{1}{2\hat{\sigma}^{2(\lambda)}} (\xi_{ik}^{(\lambda)})^2 + (\lambda - 1)\log y_i \right) \right]. \quad (2.4.23)$$

The non-parametric profile log-likelihood can then be written as

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \hat{f}_{ik}^{(\lambda)} \right). \quad (2.4.24)$$

where $\hat{f}_{ik}^{(\lambda)} = f(y_i, \lambda | \hat{z}_k)$. The non-parametric profile maximum likelihood (NPPML) estimate of λ is therefore given by

$$\hat{\lambda} = arg \max_{\lambda} \ \ell_P(\lambda). \quad (2.4.25)$$

Equation (2.4.24) is maximized over a given grid of values for λ using the values for $\hat{z}_k^{(\lambda)}$, $\hat{\beta}^{(\lambda)}$, $\hat{\sigma}^{2(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ given in Equations (2.4.15), (2.4.16), (2.4.17) and (2.4.18).

2.4.2 Estimation of the transformation parameter

In this section, we investigate the possibility of using a simpler approach that obtains the estimate of λ directly by deriving the log–likelihood with respect to λ. From equation (2.4.1), the complete log-likelihood for ($\lambda \neq 0$) can be written as

$$\ell^* = \sum_{i=1}^{n}\sum_{k=1}^{K}\Bigg[G_{ik}\log\pi_k + G_{ik}\Bigg(-\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{\sigma}\Big(\Big(\frac{y_i^\lambda - 1}{\lambda}\Big) - x_i^T\beta - z_k\Big)^2 + (\lambda-1)\log y_i\Bigg)\Bigg] \tag{2.4.26}$$

Now we differentiate equation (2.4.26) with respect to λ using $w_{ik}^{(\lambda)}$, as follows

$$\begin{aligned}
\frac{\partial \ell^*}{\partial\lambda} &= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(\frac{y_i^\lambda\log y_i}{\lambda} - \frac{y_i^\lambda - 1}{\lambda^2}\Big)\Big(\Big(\frac{y_i^\lambda-1}{\lambda}\Big) - x_i^T\beta - z_k\Big) + \log y_i\Big] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(\lambda y_i^\lambda\log y_i - y_i^\lambda + 1\Big)\Big(\lambda\Big(y_i^\lambda - 1\Big) - \lambda^2 x_i^T\beta - \lambda^2 z_k\Big) + \lambda^2\log y_i\Big] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(\lambda y_i^\lambda\log y_i - y_i^\lambda + 1\Big)\Big(\lambda y_i^\lambda - \lambda - \lambda^2 x_i^T\beta - \lambda^2 z_k\Big) + \lambda^2\log y_i\Big] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(\lambda y_i^\lambda\log y_i - y_i^\lambda + 1\Big)\Big(\lambda y_i^\lambda - \lambda - \lambda^2 x_i^T\beta - \lambda^2 z_k\Big) + \lambda^2\log y_i\Big] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(\lambda^2 y_i^{2\lambda}\log y_i - \lambda^2 y_i^\lambda\log y_i - \lambda^3 y_i^\lambda\log y_i x_i^T\beta - \lambda^3 y_i^\lambda\log y_i z_k - \lambda y_i^{2\lambda} \\
&\qquad - \lambda y_i^\lambda - \lambda^2 y_i^\lambda x_i^T\beta - \lambda^2 y_i^\lambda z_k + \lambda y_i^\lambda - \lambda - \lambda^2 x_i^T\beta - \lambda^2 z_k\Big) + \lambda^2\log y_i\Big] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\Big[-\frac{2}{\sigma}\Big(-\lambda^3 y_i^\lambda\log y_i(x_i^T\beta + z_k) + \lambda^2\Big(y_i^{2\lambda}\log y_i - y_i^\lambda(\log y_i + x_i^T\beta + z_k) \\
&\qquad - (x_i^T\beta + z_k)\Big) - \lambda(y_i^{2\lambda} + 1)\Big) + \lambda^2\log y_i\Big]
\end{aligned} \tag{2.4.27}$$

This method leads to complicated score functions and is therefore not considered further. However, solving the score function just involves root finding and a numerical approach could be used, such as `uniroot`.

2.5 Technical details

2.5.1 Non-iterative solution for $\hat{\beta}^{(\lambda)}$

Equation (2.4.15) can be plugged into Equation (2.4.16) to yield the equation of the estimate $\hat{\beta}$:

$$
\begin{aligned}
\hat{\beta}^{(\lambda)} &= \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i \left(y_i^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)} \hat{z}_k^{(\lambda)}\right) \qquad (2.5.1)\\
&= \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i \left(y_i^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)} \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}}\right)\\
&= \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i \left(\sum_{k=1}^{K} w_{ik}^{(\lambda)} y_i^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)} \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}}\right)\\
&= \left(\sum_{i=1}^{n} x_i x_i^T\right)^{-1} \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}}\right)\\
&= \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \left(y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})\right)\\
&= \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)}\\
&\quad - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} (y_m^{(\lambda)} - x_m^T \hat{\beta}^{(\lambda)})\\
&= \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)}\\
&\quad - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)}\\
&\quad + \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \hat{\beta}^{(\lambda)}\\
\Longrightarrow\ & \hat{\beta}^{(\lambda)} - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \hat{\beta}^{(\lambda)} =\\
& \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \left(y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)}\right)\\
\Longrightarrow\ & \left(I_p - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T\right) \hat{\beta}^{(\lambda)} =\\
& \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)}\right)^{-1} \left(y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)} - \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)}\right)
\end{aligned}
$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(I_p - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \right)^{-1}$$
$$\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{i=1}^{n} x_i x_i^T \sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \left(y_i^{(\lambda)} \sum_{m=1}^{n} w_{mk}^{(\lambda)} - \sum_{m=1}^{n} w_{mk} y_m^{(\lambda)} \right)$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i x_i^T - \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \right)^{-1}$$
$$\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)} \right)$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i \left(\sum_{k=1}^{K} w_{ik}^{(\lambda)} x_i^T - \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \right) \right)^{-1}$$
$$\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)} \right)$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(x_i^T - \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T \right) \right)^{-1}$$
$$\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \left(\sum_{m=1}^{n} w_{mk}^{(\lambda)} \right)^{-1} \sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)} \right)$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(x_i^T - \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \right) \right)^{-1}$$
$$\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)}}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \right)$$

$$\Longrightarrow \hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(x_i^T - \bar{x}_k^T \right) \right)^{-1} \sum_{i=1}^{n} x_i \sum_{k=1}^{K} w_{ik}^{(\lambda)} \left(y_i^{(\lambda)} - \bar{y}_k^{(\lambda)} \right) \tag{2.5.2}$$

where

$$I_p = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \tag{2.5.3}$$

and $\bar{x}_k$ and $\bar{y}_k^{(\lambda)}$ are just the weighted mean for the variables x and $y^{(\lambda)}$, respectively.

specifically,

$$\bar{x}_k = \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} x_m^T}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \quad ; \quad \bar{y}_k^{(\lambda)} = \frac{\sum_{m=1}^{n} w_{mk}^{(\lambda)} y_m^{(\lambda)}}{\sum_{m=1}^{n} w_{mk}^{(\lambda)}} \tag{2.5.4}$$

Equation (2.5.2) can be represented in matrix form as follows,

$$\hat{\beta}^{(\lambda)} = \left(\underbrace{\tilde{X}^T}_{\mathrm{P} \times \mathrm{nK}} \quad \underbrace{\tilde{W}^{(\lambda)}}_{\mathrm{nK} \times \mathrm{nK}} \quad \underbrace{(\tilde{X} - \ddot{X})}_{\mathrm{nK} \times \mathrm{P}} \right)^{-1} \underbrace{\tilde{X}^T}_{\mathrm{P} \times \mathrm{nK}} \quad \underbrace{\tilde{W}^{(\lambda)}}_{\mathrm{nK} \times \mathrm{nK}} \quad \underbrace{(\tilde{Y}^{(\lambda)} - \ddot{Y}^{(\lambda)})}_{\mathrm{nK} \times 1} \tag{2.5.5}$$

where

$$\tilde{W}^{(\lambda)} = \underbrace{\left.\begin{pmatrix} W^{(\lambda)} & \cdots & \cdots & W^{(\lambda)} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ W^{(\lambda)} & \cdots & \cdots & W^{(\lambda)} \end{pmatrix}\right\} K \text{ times}}_{n \text{ times}} \quad , \quad W^{(\lambda)} = \begin{pmatrix} w_{11}^{(\lambda)} & \cdots & \cdots & w_{1K}^{(\lambda)} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1}^{(\lambda)} & \cdots & \cdots & w_{nK}^{(\lambda)} \end{pmatrix}$$

$$\tilde{Y}^{(\lambda)} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \Big\} k = 1 \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \Big\} k = 2 \quad , \quad \begin{matrix} \vdots \\ \vdots \end{matrix} \quad \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \Big\} k = K$$

$$\tilde{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \Big\} k = 1 \quad \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \Big\} k = 2 \quad \begin{matrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{matrix} \quad \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \Big\} k = K$$

$$\ddot{Y}^{(\lambda)} = \begin{pmatrix} \bar{y}_1^{(\lambda)} \\ \vdots \\ \bar{y}_1^{(\lambda)} \\ \bar{y}_2^{(\lambda)} \\ \vdots \\ \bar{y}_2^{(\lambda)} \\ \vdots \\ \vdots \\ \bar{y}_K^{(\lambda)} \\ \vdots \\ \bar{y}_K^{(\lambda)} \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \end{matrix} \quad \text{and} \quad \ddot{X} = \begin{pmatrix} \bar{x}_{11} & \cdots & \cdots & \bar{x}_{1P} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{x}_{11} & \cdots & \cdots & \bar{x}_{1P} \\ \bar{x}_{21} & \cdots & \cdots & \bar{x}_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{x}_{21} & \cdots & \cdots & \bar{x}_{2P} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \bar{x}_{K1} & \cdots & \cdots & \bar{x}_{KP} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{x}_{K1} & \cdots & \cdots & \bar{x}_{KP} \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \\ \\ \\ \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} n \end{matrix}$$

This is essentially what we would get if we worked with the usual expanded data ($n \times K$ copies) and fitted a model with the required linear predictor and a factor for z_k with a different level for each copy. Equation (2.5.5) will be considered in the following subsection. However, in practice, we will use Equations (2.4.15) and (2.4.16) because $\hat{z}_k^{(\lambda)}$ is still needed to get the weights.

2.5.2 The standard error of the parameter estimates $SE(\hat{\beta}^{(\lambda)})$

The EM algorithm does not automatically produce standard errors of the estimates (SE). Several procedures have been proposed to compute the variance-covariance

matrices from which the SE estimates can be obtained for a low–dimensional parameter. However, these methods cannot be easily used for a large number of parameters (Xu et al., 2014). Murphy and Van der Vaart (2000) presented a semiparametric profile likelihoods technique for SE estimation which can be used whenever the infinite–dimension of the observed information presents. They pointed out that the standard error of the model parameter can be estimated via the curvature of the log profile likelihood graph which is known as the observed information. A method for computing the observed information in the context of EM has been presented in Louis (1982). The method requires computation of the second derivative matrix of the complete–data log–likelihood with respect to the model parameters and the SE estimation can be extracted directly from the EM iteration. Baker (1992) pointed out that the expected information matrix is often easier to calculate than the observed information matrix. Friedl and Kauermann (2000) derived an approximation of the variance-covariance matrix, which is based on the expected information matrix, for computing standard errors of EM algorithm for NPML estimation in generalized linear models with unknown random effects. This approximation can also be invoked at the last EM iteration.

The accuracy of estimates can be assessed using the standard error. The smaller the standard error, the more accurate the estimate. The standard error of the coefficient is a measure of the spread of the data, therefore, it can not be negative. To obtain the standard errors of the parameter estimates $SE(\hat{\beta}_i^{(\lambda)})$ in the final model, we take the square root of the diagonal elements of the variance-covariance matrix, Σ.

$$\Sigma_{i,j}^{(\lambda)} \equiv Cov(\hat{\beta}_i^{(\lambda)}, \hat{\beta}_j^{(\lambda)}) \tag{2.5.6}$$

where

$$Cov(\hat{\beta}_i^{(\lambda)}, \hat{\beta}_i^{(\lambda)}) = Var(\hat{\beta}_i^{(\lambda)}) \tag{2.5.7}$$

Refer to Equation (2.5.5), the diagonal elements of the variance-covariance matrix could be computed as follows:

$$Var(\hat{\beta}^{(\lambda)}) = Var\Big(\left(\tilde{X}^T\tilde{W}^{(\lambda)}(\tilde{X} - \ddot{X})\right)^{-1} \tilde{X}^T\tilde{W}^{(\lambda)} \left(\tilde{Y}^{(\lambda)} - \ddot{Y}^{(\lambda)}\right) \Big) =$$
$$\Big(\left(\tilde{X}^T\tilde{W}^{(\lambda)}(\tilde{X} - \ddot{X})\right)^{-1} \tilde{X}^T\tilde{W}^{(\lambda)}\, Var\left(\tilde{Y}^{(\lambda)} - \ddot{Y}^{(\lambda)}\right) \Big(\left(\tilde{X}^T\tilde{W}^{(\lambda)}(\tilde{X} - \ddot{X})\right)^{-1} \tilde{X}^T\tilde{W}^{(\lambda)} \Big)^T. \tag{2.5.8}$$

Consideration of this term has not, to our knowledge, been given attention. The analytical calculation of Equation (2.5.8) is not straightforward, however, we can approximate the standard error from the linear model equation in R by fitting the linear model given in Equation (2.5.1) which is equivalent to Equation (2.5.5) and then extract the approximate SE from it as

$$\hat{SE}(\hat{\beta}_i^{(\lambda)}) = \sqrt{\Sigma_{i,i}^{(\lambda)}} \tag{2.5.9}$$

where

$$\Sigma = s^2(X^TX)^{-1}$$

where s^2 is the error variance of the linear model fitted to the response $(Y^{(\lambda)} - W^{(\lambda)}\hat{Z}^{(\lambda)})$, and X is given in (2.3.16). Note that the standard errors of $\hat{\beta}_i^{(\lambda)}$ here are computed conditionally on λ and $\hat{z}_k^{(\lambda)}$. From this, one can calculate the t-value as

$$t\text{-value} = \frac{\hat{\beta}_i^{(\lambda)}}{SE(\hat{\beta}_i^{(\lambda)})} \tag{2.5.10}$$

That is related to a t distribution with $n-p$ degrees of freedom, where n is the sample size, p represents the number of parameters of the fitted model. The parameter estimate is statistically significant if the absolute value of the t-value is larger than 2 at the 0.05 significance level. However, the parameter significances may change once random effects are introduced (Gray, 2016). Therefore, the standard errors of the parameter estimates are needed to test the significance of the parameters. We refer the reader to Section 2.7 where we prove that the use of the approximation of the SE is usually good.

2.5.3 Starting point selection and the first cycle

In the first cycle of the algorithm, the model is fitted initially without a random effect, giving some starting values β^0 and σ^0. We now discuss in more detail the choice of starting mass points z_k^0 and corresponding masses π_k^0, for which the implementation of **boxcoxmix** allows us to choose from the two different methods as outlined below:

- Gauss-Hermite quadrature points (Einbeck and Hinde, 2006) have been described in Section 2.3.2 but the fitted model here is $y_i^{(\lambda)} = x_i^T\beta + \varepsilon_i$ and $\hat{\varepsilon}_i^{(\lambda)}$ is the residuals that are defined as the difference between the observed data of the dependent variable $y_i^{(\lambda)}$ and the fitted values $\hat{y_i}^{(\lambda)}$(i.e. $\hat{\varepsilon}_i^{(\lambda)} = y_i^{(\lambda)} - \hat{y_i}^{(\lambda)} = y_i^{(\lambda)} - x_i^T\hat{\beta}^{(\lambda)}$).

- Quantile-based version

$$z_k^{(\lambda)} = \bar{y}^{(\lambda)} + tol \times q_k^{(\lambda)} \tag{2.5.11}$$

where $\bar{y}^{(\lambda)}$ is the mean of the responses $y_i^{(\lambda)}$, tol is a positive scalar (usually,

$0 < tol <= 2$) and $q_k^{(\lambda)} = \frac{k}{K} - \frac{1}{2K}$ are quantiles of the empirical distribution of $y_i^{(\lambda)} - \bar{y}^{(\lambda)}$.

From this one obtains the extended linear predictor for the k-th component $E(y_i^{(\lambda)}|z_k) = x_i^T\beta + z_k$. Using formula (2.4.10) with current parameter estimates, one gets an "initial E-step" and in the subsequent M-step one obtains the parameter estimates by solving the score equations. From the resulting estimates of this cycle, one gets an updated value of the weights, and so on. Since we run several iterations of the EM-algorithm, one needs to stop the EM-algorithm when it reached its convergence point. Polańska (2003) defined this convergence criterion as the absolute change in the successive log-likelihood function (disparity $= -2\ell_P(\lambda)$) values being less than a certain threshold such as 0.0001.

First, we need to indicate a range over which the optimization of λ will occur. For each λ, we iterate between $\hat{\beta}^{(\lambda)}$ and $\hat{z}_k^{(\lambda)}$ in the each M-step a small number of times. And in each iteration of the EM-algorithm, we have $\hat{z}_k^{(\lambda)}, \hat{\sigma}^{2(\lambda)}, \hat{\beta}^{(\lambda)}$, $\hat{\pi}_k^{(\lambda)}$ and $w_{ik}^{(\lambda)}$ that are used together to update the NPPML ($\ell_P(\lambda)$). We end up with $\ell_P(\lambda)$ for each λ; the optimal choice for λ is the one that maximizes $\ell_P(\lambda)$. In other words, Equation (2.4.24) is maximized over a given grid of values for λ using $\hat{z}_k^{(\lambda)}, \hat{\sigma}^{2(\lambda)}, \hat{\beta}^{(\lambda)}$, $\hat{\pi}_k^{(\lambda)}$ and $w_{ik}^{(\lambda)}$.

2.6 Software description

Fitting random effect models using response transformations with an unspecified mixing distribution can be done with the R package **boxcoxmix** (Almohaimeed and

Einbeck, 2017). The main function is `optim.boxcox()` that performs a grid search over the parameter λ and then optimizes over this grid, to calculate the maximum likelihood estimator of the transformation. It produces a plot of the non-parametric profile likelihood function (2.4.25) that summarises information concerning λ, including a vertical line indicating the best value of λ that maximizes the non-parametric profile log-likelihood.

In order to fit models with fixed value of λ, one can use the function `np.boxcoxmix()` that can be used for overdispersed generalized linear models and variance component models. It produces a plot of the disparity with the iteration number on the x-axis and the mass points on the y-axis. It also produces normal Q-Q plots to determine how well a set of values follow a normal distribution. Furthermore, it plots the control charts of the residuals of the data before and after applying the transformation that are $\hat{\varepsilon}_i = y_i - \hat{y}_i = y_i - x_i^T\hat{\beta} - \hat{z}_i$ and $\hat{\varepsilon}_i^{(\lambda)} = y_i^{(\lambda)} - \hat{y}_i{}^{(\lambda)} = y_i^{(\lambda)} - x_i^T\hat{\beta}^{(\lambda)} - \hat{z}_i^{(\lambda)}$, respectively, where $\hat{z}_i = \sum_{k=1}^{K} w_{ik}\hat{z}_k$ and $\hat{z}_i^{(\lambda)} = \sum_{k=1}^{K} w_{ik}^{(\lambda)}\hat{z}_k^{(\lambda)}$ and that is to detect special causes of variation. There are many possible causes of an out-of-control point, including non-normal data and the number of classes, K.

Additionally, it produces the parameter estimates, the standard errors of the estimates, t-value and the log-likelihood value. Alternative methods of assessing model fit other than t-value are the Akaike's Information Criterion (AIC) and Bayesian Information Criteria (BIC),

$$AIC = -2\ell_P(\lambda) + 2 \times (p + 2K - 1 + c) \tag{2.6.1}$$

$$BIC = -2\ell_P(\lambda) + \log(n) \times (p + 2K - 1 + c) \tag{2.6.2}$$

where $\ell_P(\lambda)$ is the profile log-likelihood function given in (2.4.24) which is obtained by substituting the maximum likelihood estimators of the model parameters (i.e. $z_k = \hat{z}_k^{(\lambda)}$, $\pi_k = \hat{\pi}_k^{(\lambda)}$, $\beta = \hat{\beta}^{(\lambda)}$ and $\sigma = \hat{\sigma}^{(\lambda)}$), and the second part of the AIC and BIC equations computes the number of parameters estimated in the model. Note that, σ is a constant term in any given model, even though $\hat{\sigma}^{(\lambda)}$ depends on z_k and λ, this parameter σ is of no relevance for the problem of the model selection, therefore, it should not be included in the degrees of freedom (df) of the model. p is the number of regression parameters in $\hat{\beta}^{(\lambda)}$, K is the number of mixture classes, c is the value 1 if the transformation parameter is estimated and zero otherwise, and n is the number of observations (see Table 2.6.1). As such, given a set of models, the best model in terms of relative quality will be the one with minimum AIC or BIC value. Claeskens (2016) discussed the model selection process via the information criteria such as AIC and BIC in details. They remarked that "a good model should fit well and not be too complex".

Parameters	df
$\hat{z}_1^{(\lambda)}, \ldots, \hat{z}_K^{(\lambda)}$	K
$\hat{\pi}_1^{(\lambda)}, \ldots, \hat{\pi}_{K-1}^{(\lambda)}$	$K-1$
$\hat{\beta}_1^{(\lambda)}, \ldots, \hat{\beta}_p^{(\lambda)}$	p
$\hat{\lambda}$	1

Table 2.6.1: number of parameters

Skewness of the distribution of residuals can occur if the data is not normally distributed or if the number of the classes K is insufficient. For this, the function

`Kfind.boxcox()` was created to search over a selected range of K and find the best. For each number of classes, a grid search over `tol` is performed and the `tol` with the lowest AIC or BIC value is considered as the optimal. Having the minimal AIC or BIC values for a whole range of K that have been selected, the function `Kfind.boxcox()` can find the best number of components as the one with the smallest AIC or BIC value. The full range of values of K and their corresponding optimal `tol` is provided by the `Kfind.boxcox()`'s output and can be used with other **boxcoxmix** functions as arguments.

In addition, **boxcoxmix** also can be used to perform a grid search over `tol` with a fixed number of classes using the function `tolfind.boxcox()` to identify optimal starting values for the mass points, and to produce some useful diagnostic plots of objects generated by the functions `np.boxcoxmix()`, `optim.boxcox()`, `Kfind.boxcox()` and `tolfind.boxcox()`, using the generic function `plot()`. The plots to be printed depend on the choice of the argument `plot.opt`,

- 1, the disparities with the iteration number against the mass points;
- 2, the fitted values against the responses of the untransformed and the transformed data;
- 3, probability plot of residuals of the untransformed against the transformed data;
- 4, individual posterior probabilities;
- 5, control charts of residuals of the untransformed against the transformed data;

- 6, the histograms of residuals of the untransformed against the transformed data;

- 7, plots the specified range of `tol` against the disparities (works only for the function `tolfind.boxcox()`);

- 8, gives the profile likelihood function that summarises information concerning λ (works only for the function `optim.boxcox()`).

- 9, plots the specified range of K against the disparities (works only for the function `Kfind.boxcox()`);

- 10, gives the profile likelihood function that summarises information concerning λ (works only for the function `boxcoxtype()`). This function is used for logistic-type models.

Other generic functions for **boxcoxmix** are `summary()` and `print()` that print output summaries of fitted models.

When λ =1 (no transformation), the results of the proposed approach will be very similar to that of the **npmlreg** function `alldist()`. However, the function `np.boxcoxmix()` is not a copy or extension of the function `alldist()`; the implementation is based on directly computing (2.4.15)-(2.4.18) rather than relying on the output of the function `glm()`. We refer the interested reader to Appendix A.4 and Almohaimeed and Einbeck (2017) for more information on the **boxcoxmix** package.

2.7 Simulation studies

In this section, we perform two simulation studies. Each has two ways of examining the performance of the proposed approach; by using fixed values of the transformation parameters (λ) to estimate the model parameters (β), and by using unknown transformation parameters to estimate the transformation and regression parameters simultaneously. Similar to Gurka et al. (2006) and Maruo et al. (2017), we use two different designs of the simulations to determine the effect of misspecification of the the error term distribution on the estimation and inference about β and λ. The procedure used for the simulation studies is given in the appendix, Figure A.3.1.

Simulation Study 1

Firstly, we are interested in examining the method's ability to estimate the true parameter values. In order to do that, we simulate data by applying the inverse of the Box–Cox transformation given in equation (2.4.2) to a dataset that follows a normal distribution using a set of λ's values. For each value of λ_ℓ, $\ell = 1, 2, 3, 4$, we generate 1000 datasets of 100 observations as follows,

$$\zeta_{i\ell} = \hat{y}(\eta_i, \lambda_\ell), \qquad i = 1, \cdots, 100 \tag{2.7.1}$$

$$\hat{y}(\eta_i, \lambda_\ell) = \begin{cases} \left(1 + \lambda_\ell \eta_i\right)^{1/\lambda_\ell} & (\lambda_\ell \neq 0), \\ e^{\eta_i} & (\lambda_\ell = 0) \end{cases}$$

$$\eta_i = 3\, x_{1,i} + 0.5\, x_{2,i} + z_i + \varepsilon_i$$

$$X_1 \sim U(-1, 1),\ X_2 \sim U(-3, 3)$$

$$\varepsilon \sim N(0, 0.5^2)$$

$$\lambda_1 = 0, \;\; \lambda_2 = 0.5, \;\; \lambda_3 = 1, \;\; \lambda_4 = 2$$

$$z_i \sim \text{Multinomial}\{1, (z_1, \ldots, z_4)|\pi_1, \ldots, \pi_4\}$$

$$z_k = (15, 20, 30, 35) \text{ with masses } \pi_k = 1/4, \; k = 1, ..., 4.$$

To make it simple, we are going to break down the simulation process into well–defined steps in the following flowchart,

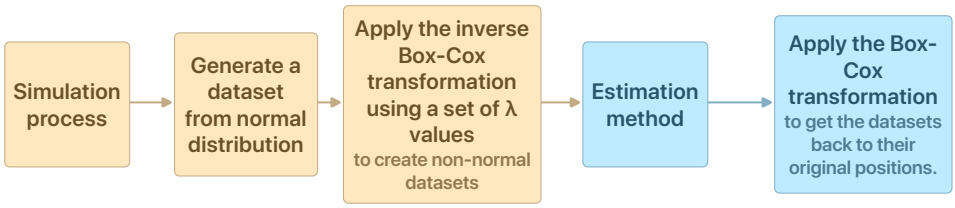


Figure 2.7.1: Flow chart of the methodology followed in the simulation study 1

First, in the simulation process, we generate a dataset from a normal distribution and then we apply inverse Box–Cox transformation denoted by $\hat{y}(\cdot)$ in (2.7.1) for λ_ℓ to create non–normal datasets using the **boxcoxmix** function `yhat()`. We end up with four datasets $\zeta_{i\ell}$ in each simulation run, where each dataset has a specific value of λ. Basically, we are trying to obtain an estimate of β that matches its true value from a data that is transformed to a normal distribution using our approach. We are also estimating the proposed standard errors of parameters $(\text{SE}(\hat{\beta}))$ to compare them with the standard deviation of the estimated β to judge their accuracy. One could consider this as a 'trivial' scenario since in effect no transformation takes place, but we consider this as a baseline test that the machinery is correctly set up. For the interested reader, Appendix A provides a brief discussion of this case together with the R code that was used to generate the simulated data.

To verify the performance of our approach, boxplots are used to compare the actual parameter values with the estimated values obtained from the simulated data. The boxplots display the range of variation in the estimated parameter values from the simulation results and also show the ability of the simulations to reproduce the actual parameter values.

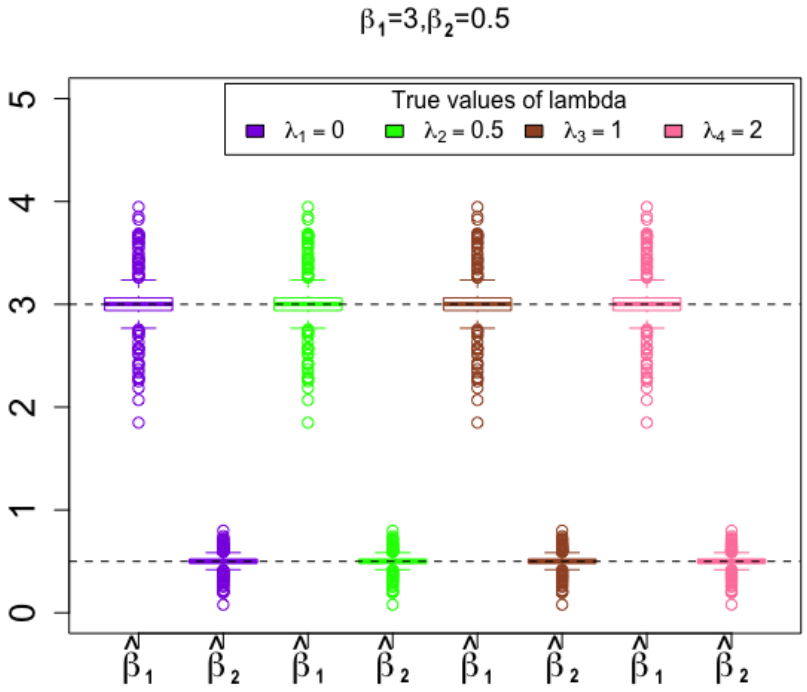


Figure 2.7.2: Simulation Study 1: boxplots for the parameter estimates of the transformed random effect model using a fixed value of λ that is 0, 0.5, 1 and 2, respectively, from 1000 simulations.

Figure 2.7.2 shows the boxplots for 1000 estimates of the transformed model parameters using $K = 4$ and a fixed value of λ that is 0, 0.5, 1 and 2, respectively. The top and bottom of each box reflect the third and first quartiles, respectively. The line in the middle of the boxes is the median of the estimated β. Two lines extend from each box to reach the maximum and minimum values. We added reference lines in the boxplots which indicate the actual values of $\beta = (3, 0.5)$, to display the

position of the estimated β for each boxplot. The parameter estimates given in Figure 2.7.2 are consistent with the true parameter values for each plot.

The interquartile range (IQR) can be used as a robust measure of the standard deviation of the estimated β since it is less influenced by outliers than the standard deviation. The IQR is the difference between the first quartile (Q_1) and the third quartile (Q_3). We can compute the IQR of the estimated β as follows

$$\text{IQR} = Q_3 - Q_1 \tag{2.7.2}$$

Via normal reference, the IQR can be mapped back to the scale of the standard deviations by division through 1.349 (Silverman, 1986). We call the resulting robust estimate of standard deviation RESD($\hat{\beta}$).

β_1	3	β_2	0.5
Mean($\hat{\beta}_1$)	3.005379	Mean($\hat{\beta}_2$)	0.5000106
Median($\hat{\beta}_1$)	3.002567	Median($\hat{\beta}_2$)	0.5010484
RESD($\hat{\beta}_1$)	0.09044292	RESD($\hat{\beta}_2$)	0.03115203
Mean($\hat{SE}(\hat{\beta}_1)$)	0.1128143	Mean($\hat{SE}(\hat{\beta}_2)$)	0.037569
Median($\hat{SE}(\hat{\beta}_1)$)	0.08587615	Median($\hat{SE}(\hat{\beta}_2)$)	0.02858872

Table 2.7.1: Simulation Study 1: Summary of simulation results for $\lambda = 0$

Table 2.7.1 displays RESD($\hat{\beta}$) values along with means and medians of EM–based standard errors, $\hat{SE}(\hat{\beta})$, which were obtained by extraction from the model fitted in the last M–step refer to Section (2.5.9). The first row is the true values of

β from which the simulation data sets were generated. Since the results for all λ's values are identical in this case, only the results of one value of λ are presented. It is conceptually clear that such EM–based standard errors cannot be 'correct' as they ignore the variation caused by the EM algorithm itself, but we see from Table 2.7.1 that they are still satisfyingly close to their empirical counterparts.

Next, we return to the earlier simulation design Equation (2.7.1) to test the ability of our approach to estimate the transformation and regression parameters simultaneously. The simulation process is the same and only the estimation method is diffrent. Here we estimate λ by applying a grid search over λ for $\zeta_{i\ell}$ given in Equation (2.7.1) via the function `optim.boxcox()` and estimate β together with the SE($\hat{\beta}$) using the optimal value of λ. Importantly, the design of the simulation affects the final results so before running the simulations we need to choose the covariates X and the starting points z_k, K, β and σ, carefully (refer to Appendix A for details).

As we mentioned beforehand we are interested in transforming the data to be normal or close to normal. Now we apply the backward transformation to η_i using a fixed value of λ that is 0, 0.5, 1 and 2, respectively. In the next step, we transform the simulated data ($\zeta_{i\ell}$) to bring it back to the normal distribution using the **boxcoxmix** function `optim.boxcox()` which uses a range of λ's values to estimate the optimum λ. The aim here is to get estimates of λ and β that are approximately equal to the actual values using fixed value of K that was used in the simulation step ($K = 4$). Thus, we repeat the simulation for 1000 times to obtain 1000 estimates of the optimum λ and their corresponding estimates of β and SE($\hat{\beta}$) at once. From this, we obtain the median of the estimated parameters using the

boxplots and compare it with their true values.

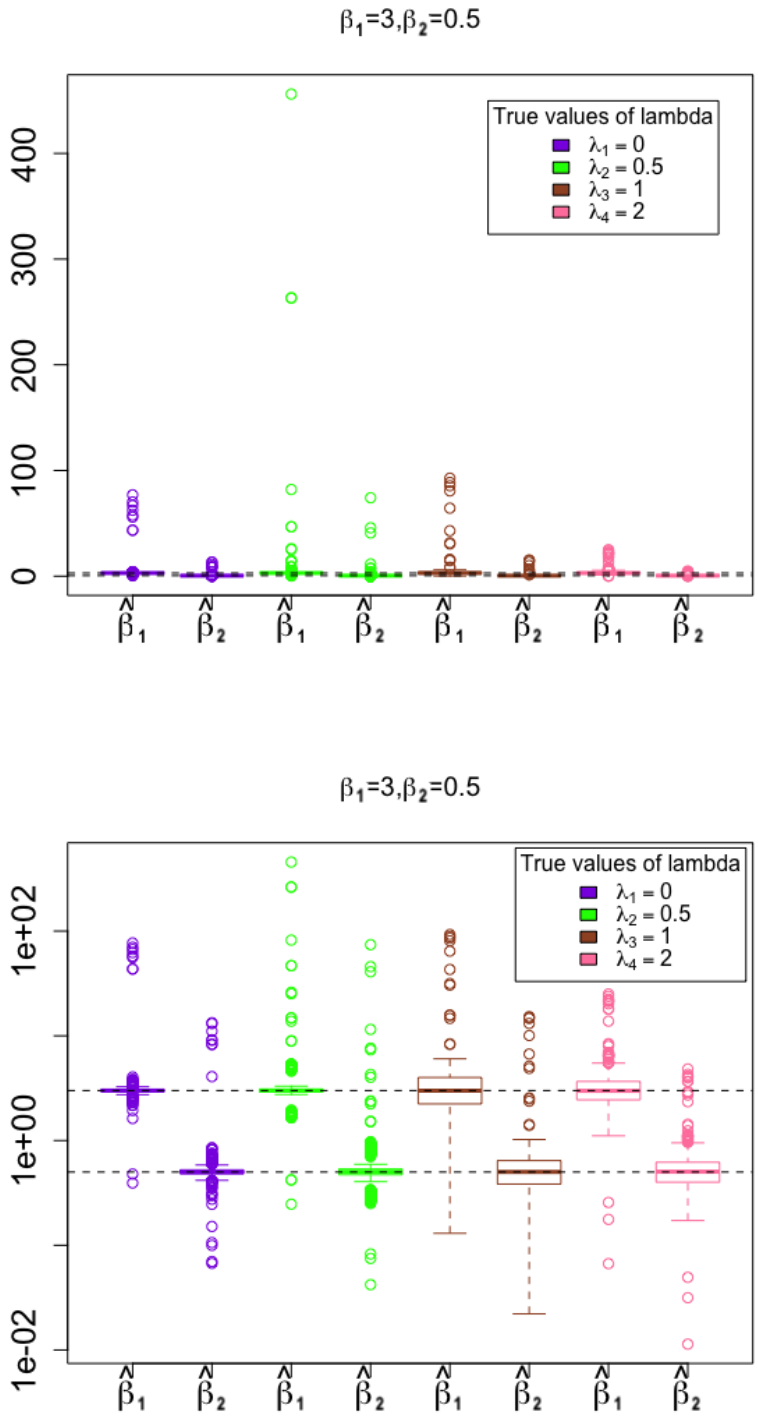


Figure 2.7.3: Simulation Study 1: estimates $\hat{\beta}$, in each plot for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right). The lower plot is exactly the upper plot with logarithmic scale in the vertical axis. Horizontal lines indicate the true values.

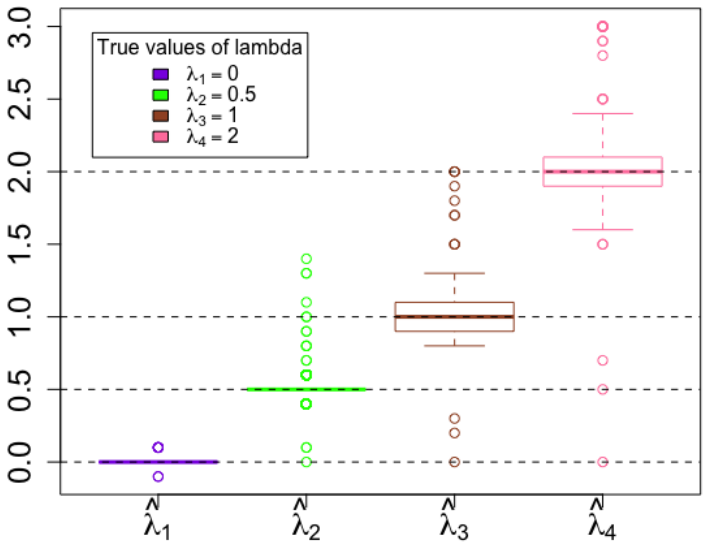


Figure 2.7.4: Simulation Study 1: estimated λ, for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right).

Figure 2.7.3 shows the boxplots for the parameter estimates, for each model transformed by the optimal λ that was obtained after applying a grid search over λ where the true $\lambda = 0, 0.5, 1, 2$. The horizontal lines in the boxplots indicate the actual values of $\beta = (3, 0.5)$. It is clear that the median of the estimated β is close to the true value in each plot. The boxplots for the estimated values of λ is plotted in Figure 2.7.4 for each transformed model using a grid search over λ for $\zeta_{i\ell}$ where the actual value of λ are $0, 0.5, 1$ and 2. We added reference lines which perform the actual values of λ. One can see that the median of the estimated λ matches the true value in each plot. The medians of the estimated β and λ parameters are also provided in Table 2.7.2; we see that the medians for the regression parameters approximately equal the actual parameter values, and those of the transformation parameters are exactly equal to their true values.

Table 2.7.2 displays the mean and median of the estimated standard errors of the regression parameters together with the RESD($\hat{\beta}$). Column value of λ and row value of β are considered the true values from which the simulation data sets were generated.

	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
Mean($\hat{\lambda}$)	0.0006	0.5057	1.0046	2.0013
Median($\hat{\lambda}$)	0	0.5	1	2
β_1	3	3	3	3
Mean($\hat{\beta}_1$)	3.44953	4.348057	3.629167	3.307425
Median($\hat{\beta}_1$)	3.001169	3.002301	2.991664	2.981476
RESD($\hat{\beta}_1$)	0.09800532	0.1393788	1.304201	0.9044531
Mean($\hat{SE}(\hat{\beta}_1)$)	0.1428246	0.1394787	0.1185957	0.1031293
Median($\hat{SE}(\hat{\beta}_1)$)	0.08647465	0.08530036	0.08469066	0.08446234
β_2	0.5	0.5	0.5	0.5
Mean($\hat{\beta}_2$)	0.5742274	0.7197623	0.604555	0.5541792
Median($\hat{\beta}_2$)	0.5011232	0.5027145	0.5017513	0.5042211
RESD($\hat{\beta}_2$)	0.03227793	0.04796036	0.1920292	0.1632745
Mean($\hat{SE}(\hat{\beta}_2)$)	0.04739289	0.04663437	0.03962645	0.03443026
Median($\hat{SE}(\hat{\beta}_2)$)	0.02874361	0.02825081	0.02810237	0.02809227

Table 2.7.2: Simulation Study 1: Summary of simulation results using $\hat{\lambda}$, in each column for true $\lambda_\ell = 0, 0.5, 1, 2$

As shown in Table 2.7.2, RESD($\hat{\beta}$) differs from the median and the mean of $\hat{SE}(\hat{\beta})$ for $\lambda = 1$ and $\lambda = 2$, however, for a smaller value of λ the the median and the mean of $\hat{SE}(\hat{\beta})$ come closer to the RESD($\hat{\beta}$) value. A closer look at the boxplots in Figure 2.7.4 shows that the variation around the true value increases

as λ gets larger and that causes the variability of the parameter estimates, yielding biases of the estimated standard errors of the parameters for larger values of λ. As mentioned before, the EM–based standard errors ignore the variation caused by the EM algorithm itself. Hence, this aspect of variation of the parameter estimates here was ignored, leading to wrong standard errors for larger values of λ.

Simulation Study 2

We investigate the effects of the varying structures of the simulated dataset on the estimation method. In this case, the simulation process and the estimation method are the same as in the previous study. We only replace the design of generated data in (2.7.1) by the following design,

$$\zeta_{i\ell} = \hat{y}(\eta_i, \lambda_\ell), \qquad \ell = 1, ..4, \;\; i = 1, ..100 \tag{2.7.3}$$

$$\hat{y}(\eta_i, \lambda_\ell) = \begin{cases} \left(1 + \lambda_\ell \eta_i\right)^{1/\lambda_\ell} & (\lambda_\ell \neq 0), \\ e^{\eta_i} \qquad (\lambda_\ell = 0) \end{cases}$$

$$\eta_i = 5x_{1,i} + 3x_{2,i} + z_i + \varepsilon_i$$

$$X_1 \sim U(-1, 1), \; X_2 \sim U(0, 4)$$

$$\varepsilon \sim N(0, 0.5^2)$$

$$\lambda_1 = 0, \;\; \lambda_2 = 0.5, \;\; \lambda_3 = 1, \;\; \lambda_4 = 2$$

$$z_i \sim \text{Multinomial}\{1, (z_1, \ldots, z_4) | \pi_1, \ldots, \pi_4\}$$

$$z_k = (15, 20, 30, 35) \text{ with masses } \pi_k = 1/4, \; k = 1, ..., 4.$$

In this case, the same error and random effects distributions that have been used in the previous study were employed and the only differences between these two

simulation studies are the true values of the coefficient β and the distribution of the second covariate X_2. In these studies, the residuals are expressed as $\hat{\varepsilon}_i^{(\lambda)} = \eta_i^{(\lambda)} - \hat{\eta}_i^{(\lambda)} = \eta_i^{(\lambda)} - x_i^T \hat{\beta}^{(\lambda)} - \hat{z}_i^{(\lambda)}$ where $\hat{z}_i^{(\lambda)} = \sum_{k=1}^{K} w_{ik}^{(\lambda)} \hat{z}_k^{(\lambda)}$. Note that, for this case, the combination of the distributions from residuals, discrete random effects and the covariates did not lead to a normal distribution of residuals. The residuals plots from the fit of the model to the simulated data in their original forms (i.e. without transformation) for these two studies are shown in Appendix A for comparison. Now we repeat the simulation for 1000 times for each fixed value of λ to obtain 1000 estimates of β using our approach with the same value of λ that was used to generate the data set.

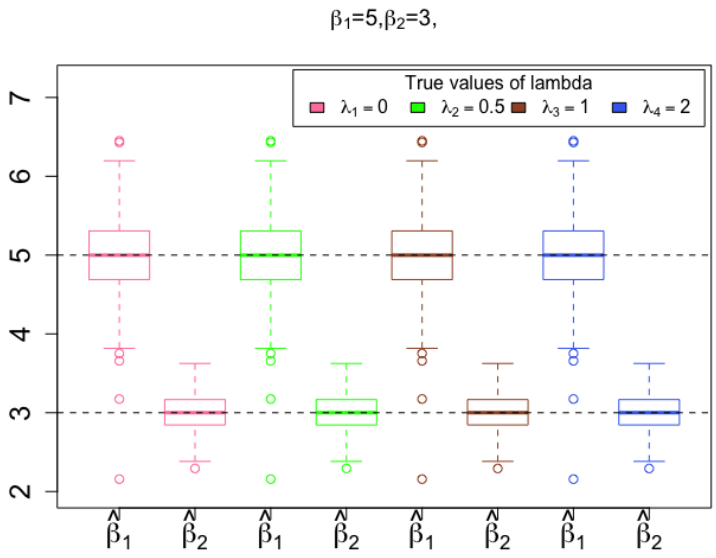


Figure 2.7.5: Simulation Study 2: boxplots for the parameter estimates of fixed lambda, for each transformed model using the true value of lambda 0, 0.5, 1 and 2, respectively, from 1000 simulations.

Figure 2.7.5 shows the boxplots for the parameter estimates, for a fixed value of $\lambda = 0$, 0.5, 1 and 2, respectively, with $K = 4$. We added the actual values of $\beta = (5, 3)$ as dotted lines in the boxplots. The median of the parameters being

estimated is close to the true value, although the estimates have some variations around the true values. However, the boxplots of the previous study presented in Figure 2.7.2 showed a stronger consistency of the parameter estimates.

As in the previous simulation study, we generate 1000 datasets as in (2.7.3) for each λ to obtain 1000 estimates of β and λ simultaneously by applying a grid search over λ seting $K = 4$. In the following graphs, we attempt to illustrate how the design of the simulated data impact the estimation results. Figure 2.7.6 shows the boxplots of the parameter estimates for each model transformed by the optimal λ that were obtained after applying a grid search over λ for $\zeta_{i\ell}$ where the true values of λ are 0, 0.5, 1 and 2. Again, the horizontal lines in the boxplots indicate the actual values of the parameters. It is clear that the medians of the estimated β are much further from the true value for $\hat{\lambda}_2, \hat{\lambda}_3$ and $\hat{\lambda}_4$. In contrast, the median of the estimated β is close to the actual value for $\hat{\lambda}_1$. The same manner is seen in Figure 2.7.7 which shows the boxplots for the transformation parameters estimates, only the median of $\hat{\lambda}_1$ captured the actual value of 0, whereas the medians of the rest of the boxplots are far from the actual values of each λ_ℓ, $\ell = 2, 3, 4$. By taking another look at the boxplots in Figures 2.7.6 and 2.7.7, we notice that the bias in $\hat{\lambda}$ causes the bias in $\hat{\beta}$. The reason for this bias appearing for $\lambda \neq 0$ lies in the fact that the η_i were originally not normally distributed due to the impact of the discrete distribution for the random effects and the distributions of the covariates on the underlying distribution of the residuals. Thus, if we transform them backwards followed by a forwards transformation using the same value of λ, we would bring them back to the non-normal distribution which is not the way that our approach actually works. Therefore, our approach selected the best estimates of λ that are

as expected different than the true values towards transforming the data into a closer-to-normal distribution.

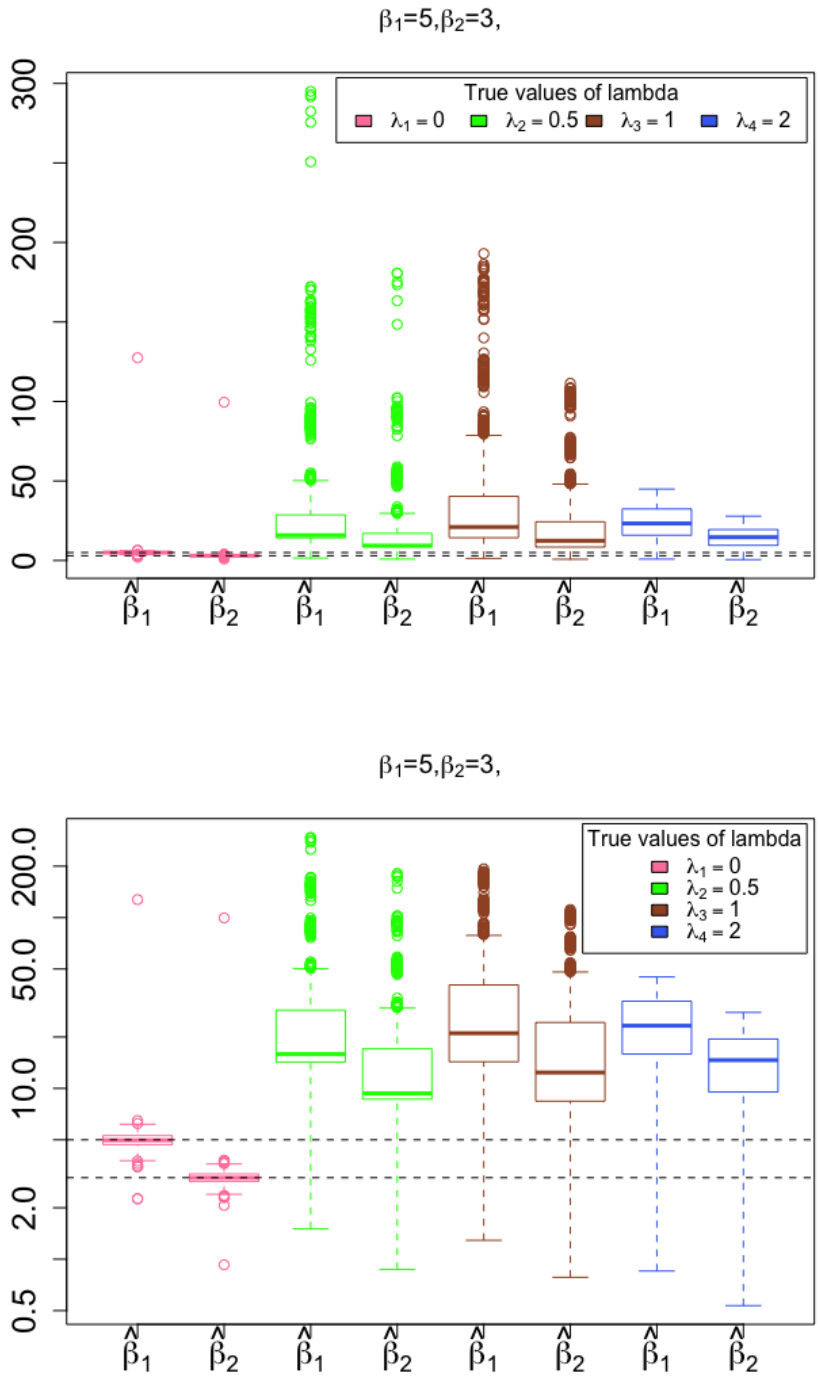


Figure 2.7.6: Simulation Study 2: estimates $\hat{\beta}$, in each plot for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right). The lower plot is exactly the upper plot with logarithmic scale in the vertical axis. Horizontal lines indicate the true values.

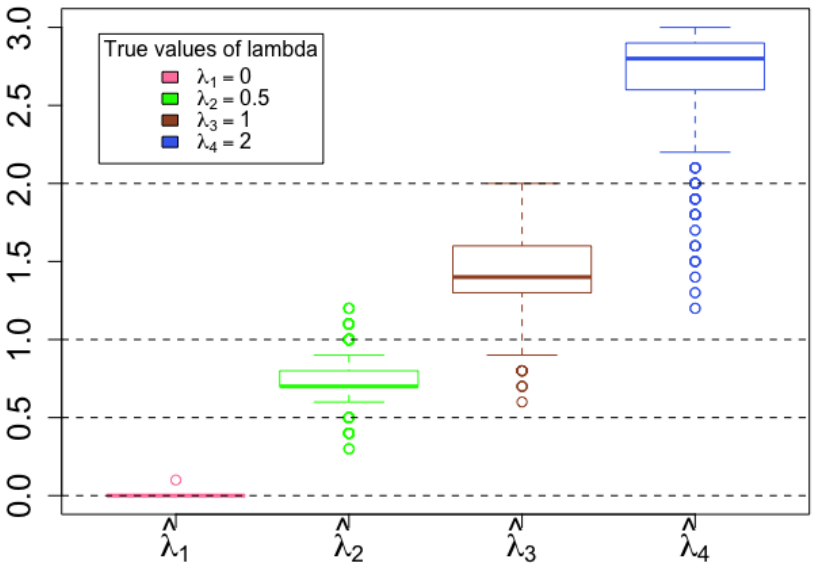


Figure 2.7.7: Simulation Study 2: estimated λ, for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right).

2.8 To transform or not to transform?

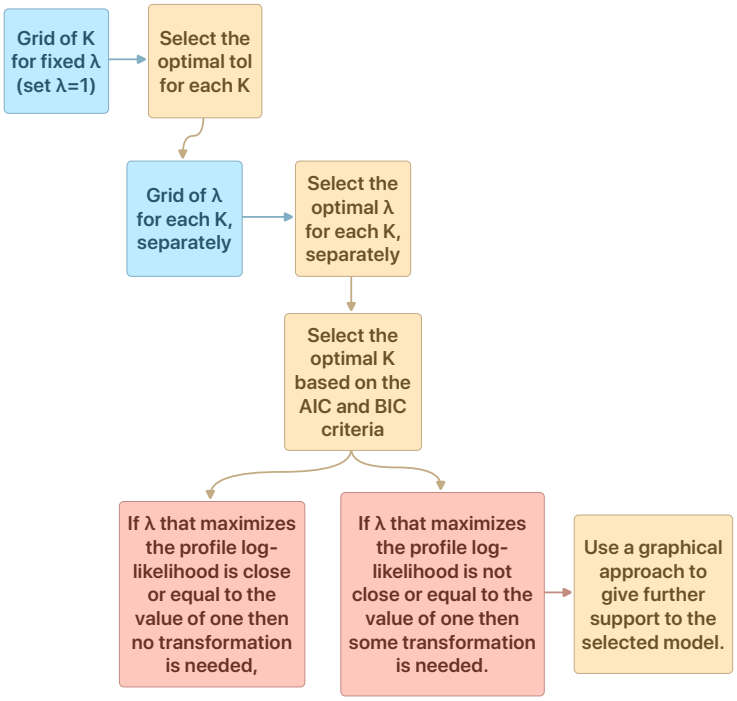


Figure 2.8.1: to transform or not to transform

Figure 2.8.1 shows the flow chart for the choice of the 'best' model in terms of which to transform data or not to transform. First, we find the optimal `tol` for each number of classes K setting $\lambda = 1$ (i.e. no transformation) where the optimal `tol` is the one that minimizes the AIC or BIC value and then perform a grid search over λ for each K with its corresponding `tol`. As we mentioned earlier, the NPML estimation may require an unnecessarily high number of components to maximize the likelihood whereas well-fitting models with a small number of components are usually preferred (Leroux and Puterman, 1992). Therefore, we use AIC and BIC together in model selection to find a model that is favoured by both criteria or is favoured by one of them but it has a fewer number of classes. If one is, furthermore, uncertain about whether to transform the data or not to transform, use the available graphical measures such as control charts, probability plots, histograms of residuals, and plots of the fitted values against the response of the untransformed and the transformed data.

2.9 Applications

Example 2.9.1. the Strength data

We have already fitted the random effects model to the `strength` data (see Example 2.3.1). We extend this analysis to the Box–Cox transformation. Again, the objective here is to investigate the effects of the covariates `lot` and `cut` on the impact strength. The random effect model that is fitted to the `strength` data is as follows,

$$y_{ij}^{(\lambda)} = \gamma_i + \beta_j + \delta_{ij} + z, \qquad i = 1, 2, \quad j = 1, 2, .., 5, \tag{2.9.1}$$

where $\gamma_1 = 0$, $\beta_1 = 0$, $\delta_{1,1} = \delta_{1,2} = \cdots = \delta_{1,5} = \delta_{2,1} = 0$, and z is the random effect with an unspecified mixing distribution, $g(z)$. To obtain initial guesses for z_k and π_k we used the Gauss-Hermite quadrature points method.

Shuster and Miura (1972) considered Inverse Gaussian distribution as adequate distribution in modelling **strength** data. We therefore suggest to fit a number of models including the Inverse Gaussian model and compare the results using the Akaike Information Criteria (AIC) defined in Equation (2.6.1). The model with the lowest AIC value is considered as the best. Here we use a three–component mixture model for all fitted models because it is the maximum number of classes that we can use for this data with the Inverse Gaussian family of the model function given in (2.9.1) using `alldist()` function; otherwise, `alldist()` will output an error message. And of course, we have compared the one and two–component mixture models with the three–component mixture model and the latter has the lowest values of disparity and AIC. For the starting point selection, the optimal value of `tol` is selected using a grid search over `tol` using **boxcoxmix** function `tolfind.boxcox()` (see Figure 2.9.1).

R Note:

Perform a grid search over `tol` for the random effect model,

```
library(boxcoxmix)
maxtol <- tolfind.boxcox(y ~ cut*lot, data = strength,
                     K = 3,   start = "gq" , lambda=1)
# Minimal Disparity: -86.61931 at tol= 1.8
# Minimal Disparity with EM converged: -86.61931 at tol= 1.8
```

```
plot(maxtol, 7)
```

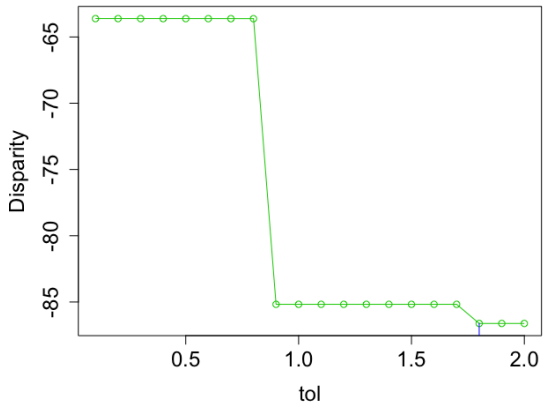


Figure 2.9.1: A grid search over *tol* for the random effect models of the `strength` data, using $K = 3$ and $\lambda = 1$

From Figure 2.9.1, one could state that the disparity of the fitted model varies continuously in the specified range of tol (from zero to two) with a minimum disparity value of -86.61931 at `tol`$= 1.8$. Using now our grid search method `optim.boxcox()` that calculates and plots the non–parametric profile log–likelihood values for the fitted model (2.9.1) against a set of λ values, and locates the NPPML of $\hat{\lambda}$ (see Figure 2.9.2):

R Note:

Perform a grid search over λ for the random effect model,

```
maxlambda <- optim.boxcox(y ~ cut*lot, data = strength,
                      K = 3, tol = 1.8,  start = "gq")
#Maximum profile Log-likelihood: 49.01121 at lambda= 0.1


plot(maxlambda,8)
```

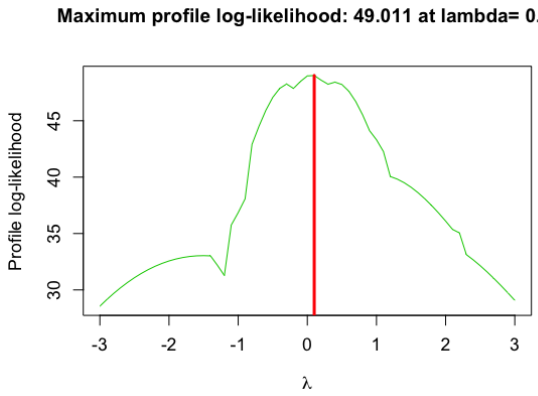


Figure 2.9.2: A grid search over λ for the random effect models of `strength` data, using $K = 3$ and `tol`$= 1.8$

Figure 2.9.2 shows that the best value of λ that maximizes the non–parametric profile log-likelihood is 0.1 which is close to zero, suggesting that some transformation need to be carried out to make the data distribution appears more normal. Now, we fit the Inverse Gaussian model using the **npmlreg** function `alldist()` with the optimal value of `tol` selected using the **npmlreg** function `tolfind()`. For a fixed value of λ, we use the **boxcoxmix** function `np.boxcoxmix()` with $\lambda = -1$ for the reciprocal transformed model and $\lambda = 1$ for the untransformed model.

R Note:

Fit the random effect model with the Inverse Gaussian family,

```
library(npmlreg)
invgauss <- alldist(y ~ cut*lot, data = strength,
                k = 3,  tol=0.45, family = "inverse.gaussian")
```

R Note:

Fit the random effect model with fixed value of λ, $\lambda = 1$ and $\lambda = -1$, respectively, where $\lambda = 1$ means no transformation is applied

```
lambda1 <- np.boxcoxmix(y ~ cut*lot, data = strength,
                  K = 3, tol = 1.8, start = "gq", lambda=1 )


lambdaneg1 <- np.boxcoxmix(y ~ cut*lot, data = strength,
                  K = 3, tol = 1.8, start = "gq", lambda=-1 )
```

	Inv.Gauss	$\lambda = -1$	$\hat{\lambda} = 0.1$	$\lambda = 1$
γ_2	0.3611	-0.4174	-0.2943	-0.2555
β_2	-0.3280	-0.1310	-0.0887	-0.0801
β_3	0.4435	-0.4522	-0.3175	-0.2722
β_4	0.0857	-0.0338	-0.2383	-0.2203
β_5	2.2516	-0.8161	-0.6845	-0.5401
$\delta_{2,2}$	-0.5111	0.4965	0.3715	0.3323
$\delta_{2,3}$	0.5146	0.1813	0.1141	0.1554
$\delta_{2,4}$	-0.1999	0.3404	0.4604	0.4070
$\delta_{2,5}$	-0.1923	0.2595	0.3378	0.3536
σ	0.3966	0.06169	0.0207	0.0206
$-2\ell_P(\lambda)$	-68	-73.70853	-98.02242	-86.61931
`AIC`	-40	-45.7085	-68.02242	-58.6193

Table 2.9.1: Comparison of results from untransformed & transformed `strength` data, using $K = 3$.

Table 2.9.1 displays summary statistics for the Inverse Gaussian distribution model (Inv.Gauss), the transformed model using $\lambda = -1$ and $\hat{\lambda} = 0.1$, and the untransformed model ($\lambda = 1$). Note that comparing the coefficients makes no sense

since the estimates of β vary greatly with a very minor change of choice of $\hat{\lambda}$. Also, the Inverse Gaussian model is not the same as a reciprocal transformation model ($\lambda = -1$), as it is clear from the subsequent output. The Inverse Gaussian model gives the worst AIC. Better AIC values are given by the transformed model using $\lambda = -1$, the Gaussian ($\lambda = 1$) and $\hat{\lambda}$. The lowest AIC found was for the transformed model using $\hat{\lambda}$ with -68.0224. The parameter estimates of the untransformed and the Box–Cox–transformed model using $\hat{\lambda}$ are in agreement but the latter has better disparity and AIC values. However, the results from the other models are quite different and the largest disparity value was founded for the Inverse Gaussian model.

K	$\lambda = -1$	$\lambda = 0.1$	$\lambda = 1$
1	-30.01438	-33.57915	-29.45051
2	-50.10725	-56.71019	-44.64449
3	-45.70853	-70.02242	-58.61931
4	-50.42968	-59.40018	-52.4271
5	-57.4437	-60.17015	-49.17725
6	-64.53892	-51.40021	-44.42724
7	-49.44363	-52.17016	-54.39248

Table 2.9.2: Comparison of AIC values for `strength` data

The appropriate number of classes K given `tol`= 1.8, could be obtained by comparing the AIC from fitting several mixture models with different numbers of classes K. Among the four models above, the one with $\lambda = 0.1$ provides the best fit of the data with $K = 3$ (see Tables 2.9.1 and 2.9.2), which does not necessarily support

the model choice taken in Shuster and Miura (1972). In the case of fixed effect model, the Box–Cox transformation for this data suggests that a transformation is needed and the natural log transformation would be appropriate.

Example 2.9.2. Fabric data

In this example, we consider a data set available as part of the R package **npmlreg** (Einbeck et al., 2014), which consists of 32 observations. The data set is analyzed by McLachlan and Peel (2004) and Aitkin et al. (2005) using NPML estimates for two and three mass-points for the Poisson mixture regression model. Furthermore, Aitkin (1996a), Hinde and Demétrio (2007) and Einbeck and Hinde (2009) fitted several overdispersion models to this data and compared their results with those for Poisson/non–parametric mixture model.

We are interested in the effect of the number of faults in rolls of fabrics `y` on the log of the length of the roll given by the variable `x`. For comparison, we apply the transformation for fixed and random effects models. The fixed effect model of interest for the `fabric` data is as follows

$$y_i^{(\lambda)} = \beta_0 + \beta_1 \cdot \mathtt{x}_i \tag{2.9.2}$$

For random effect model, a random effect z_i with an unspecified mixing distribution $g(z)$ is added to the linear predictors. That is

$$y_i^{(\lambda)} = \beta_1 \cdot \mathtt{x}_i + z_i. \tag{2.9.3}$$

Again, in order to select the appropriate number of classes, the model in (2.9.3) is fitted with $\lambda = 1$ for a set of K values, $K \in [2, 8]$. The optimal `tol`'s values, the disparities, AIC and BIC values for each K are given in Table 2.9.3, where K=1

refers to the fixed effect model given in (2.9.2). From Table 2.9.3, one could state that the AIC and BIC values of the untransformed data ($\lambda = 1$) varies continuously in the specified range of K (from one to eight) with minimum AIC and BIC values at $K = 1$.

K	1	2	3	4	5	6	7	8
optim `tol`	–	1.5	1.5	1.5	1.4	0.1	0.1	0.1
$-2\ell_P(\lambda)$	194.2763	192.2114	192.2114	192.2114	192.2114	192.2112	192.2096	181.1997
AIC	200.2763	202.2114	206.2114	210.2114	214.2114	218.2112	222.2096	215.1997
BIC	204.6735	209.5401	216.4715	223.4030	230.3345	237.2658	244.1957	240.1172

Table 2.9.3: Comparison of results from the untransformed `fabric` data ($\lambda = 1$), using K from 1 to 8

The AIC and BIC values of the model after applying the response transformation for K from 1 to 8 are shown in Figure 2.9.3 and Table 2.9.4, while Figure 2.9.4 plots $\hat{\lambda}$ as a function of K with the optimal `tol` for each number of classes.

K	1	2	3	4	5	6	7	8
$\hat{\lambda}$	0.1	-0.3	-0.3	-0.3	-0.3	-0.4	-0.4	-2.8
$-2\ell_P(\hat{\lambda})$	175.6536	171.8758	171.8758	171.8758	171.8757	164.9376	162.3069	142.5834
AIC	183.6536	181.8758	185.8758	189.8758	193.8757	190.9376	192.3069	176.5834
BIC	189.5166	189.2044	196.1360	203.0674	209.9988	209.9922	214.2930	201.5009

Table 2.9.4: Comparison of results from the transformed `fabric` data using $\hat{\lambda}$, using K from 1 to 8

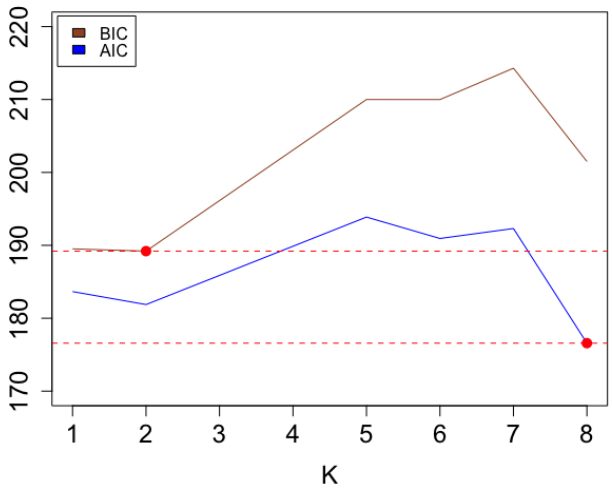


Figure 2.9.3: AIC and BIC values of the model after applying the response transformation to the `fabric` data for $K \in [1, 8]$

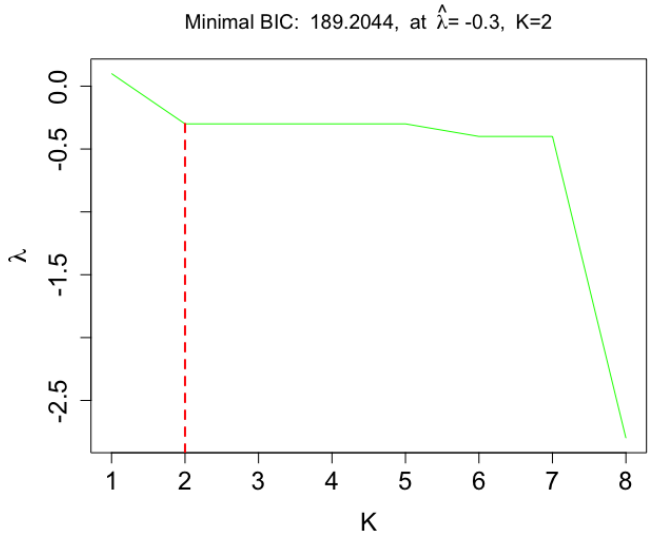
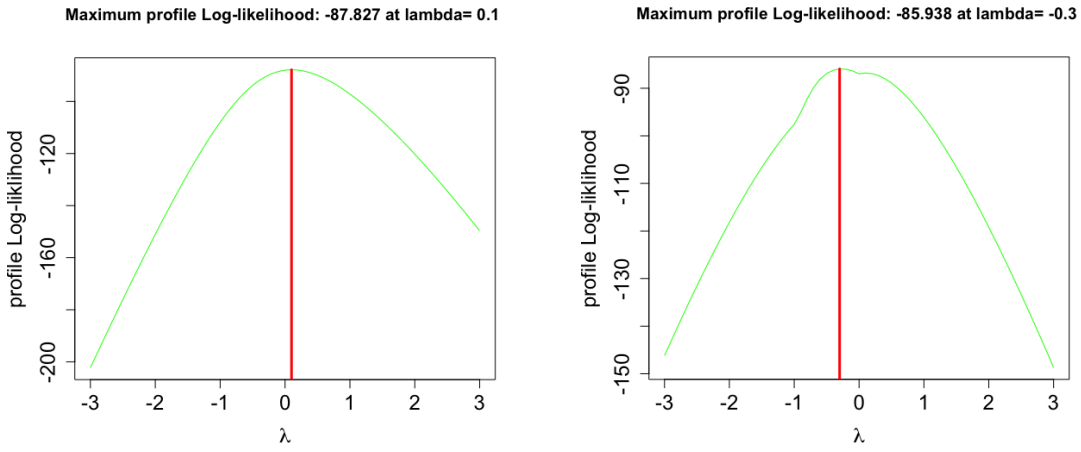


Figure 2.9.4: $\hat{\lambda}$ as a function of K with the optimal `tol` of each class for modelling `fabric` data

The minimal AIC value occurred at $K = 8$ with $\lambda = -2.8$ (AIC=176.5834), while the minimal BIC value occurred at $K = 2$ with $\lambda = -0.3$ (BIC=189.2044), see Figure 2.9.3 and Table 2.9.4. In this example, the two–component transformed model amongst the considered models is selected as the 'best model'. Figure 2.9.4 shows a strong need of a transformation as we increase the number of classes. That provides

additional evidence for a better fit to the transformed data. Figure 2.9.5 shows the Box–Cox transformation for the fixed and random effects models. It can be seen that the best estimate of λ that maximizes the profile log–likelihood for fixed effect model is 0.1 as shown in Figure 2.9.5(a) while λ that maximizes the non–parametric profile log–likelihood for random effect model in Figure 2.9.5(b) is -0.3, suggesting that both models need to be transformed to make the data distributions look more normal.



(a) transformed fixed effect model

(b) transformed random effects with K=2 and tol=1.5

Figure 2.9.5: the Box–Cox transformation for the fixed (left) and random effects models (right) to the `fabric` data

The comparison of the fitted values against the transformed response plots for fixed effect model (K =1) with $\lambda = 0.1$ to those of random effect model (K =2) with $\lambda = -0.3$ in Figure 2.9.6 demonstrates the importance of adding the random effect, where it is shown that the widely spread points of the fixed effect model become closer to a straight line. This plot can also be used as an alternative method for selecting the number of classes of the model. Aitkin (1996a), McLachlan and Peel (2004) and Einbeck and Hinde (2009) suggested using only two mass–points for

fitting Poisson mixture model with NPML to this data.

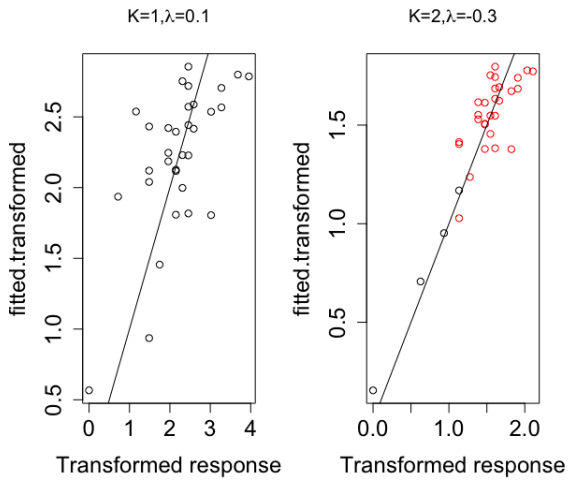


Figure 2.9.6: The fitted values against the transformed response of the `fabric` data for fixed effect model (left) and those for random effect model (right)

Figure 2.9.7: Control Chart of residuals of the untransformed (top plot) against the transformed `fabric` Data (bottom plot), using K=2, $\lambda = -0.3$ and tol=1.5

The control charts can be used as a tool to assess the normality of the data

and/or the homogeneity of variance. There are many possible causes of an out of control point, including non-normal distribution and/or non-constant variance. The control charts of residuals of the data before and after applying the transformation using $\lambda = -0.3$, with K=2 are shown in Figure 2.9.7, the top plot shows an out-of-control point beyond the control limits of the untransformed data. Another look at the control charts reveals that the points of the transformed data (the bottom plot) are much closer to the centerline of the chart than those of the untransformed data, that shows some variance stabilisation. This supplies the evidence that the transformed random effect model is more appropriate for this data.

For K =2, where $\hat{\lambda}$ is close to zero, the log transformation would be a more natural choice, since it would correspond to $Var(Y) \approx \mu^2$ which is quite compatible with many overdispersed Poisson distributions, such as the negative binomial that fit these data well. It is also what we get if we take a Poisson log-linear model and include a random effect in the linear predictor — the marginal variance is quadratic and so a log-transformation can work well. While for a standard Poisson model (not overdispersed) a square-root transformation would be variance stabilising.

2.10 Special case: Box-Cox transformations for pure mixture model

In this section, the Box-Cox transformation is adapted to the random effect model without any independent variables (we call it a 'pure mixture model' to distinguish

it from a more general type of mixture model 'mixed effect model'). Recall the equation for the Box-Cox transformation of the response y_i above

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_i & (\lambda = 0) \end{cases} \tag{2.10.1}$$

and that for $y_i > 0$, $i = 1, ..., n$. The aforementioned approach is carried out to estimate the likelihood in the same way as the random effect models with some related changes.

2.10.1 Estimation of finite mixtures

In the case of pure mixture model, it is assumed that there is a value of λ for which,

$$y_i^{(\lambda)} | z_i \sim N(z_i, \sigma^2) \tag{2.10.2}$$

where z_i is again unspecified. Taking account of the Jacobian of the transformation from y to $y^{(\lambda)}$, the conditional probability density function of y_i given z_i is

$$f(y_i, \lambda | z_i) = \frac{y_i^{\lambda - 1}}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} (y_i^{(\lambda)} - z_i)^2 \right] \tag{2.10.3}$$

The likelihood can again be approximated using NPML approach as in equation (2.4.5), yielding the log-likelihood

$$\ell = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k f_{ik}^{(\lambda)} \right) \tag{2.10.4}$$

where $f_{ik}^{(\lambda)} = f(y_i, \lambda | z_k)$. Refer to (2.3.7) and (2.4.7), the complete log-likelihood would be

$$\ell^* = \sum_{i=1}^{n} \sum_{k=1}^{K} \left[G_{ik} \log \pi_k + G_{ik} \log f_{ik}^{(\lambda)} \right] \tag{2.10.5}$$

where

$$\begin{aligned} \log f_{ik}^{(\lambda)} &= \log \left(\frac{y_i^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{1}{2\sigma^2} (y_i^{(\lambda)} - z_k)^2 \right] \right) \\ &= \left(-\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (y_i^{(\lambda)} - z_k)^2 + (\lambda - 1) \log y_i \right), \end{aligned} \tag{2.10.6}$$

then

$$\begin{aligned} \ell^* = \sum_{i=1}^{n} \sum_{k=1}^{K} \Bigg[G_{ik} \log \pi_k + G_{ik} \Bigg(& -\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (y_i^{(\lambda)} - z_k)^2 \\ & + (\lambda - 1) \log y_i \Bigg) \Bigg]. \end{aligned} \tag{2.10.7}$$

Applying the EM approach to approximate the MLE of the model parameters:

E-step: This is identical to (2.4.10), but $f_{ik}^{(\lambda)}$ here is as in (2.10.3).

M-step: Calculate $\hat{z}_k^{(\lambda)}, \hat{\sigma}^{2(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ using current $w_{ik}^{(\lambda)}$,

$$\begin{aligned} \frac{\partial \ell^*}{\partial z_k} = -\frac{1}{2\sigma^2}(2) \sum_{i=1}^{n} w_{ik}^{(\lambda)} (y_i^{(\lambda)} - z_k)(-1) &= 0 \\ \sum_{i=1}^{n} w_{ik}^{(\lambda)} (y_i^{(\lambda)} - z_k) &= 0 \\ \sum_{i=1}^{n} w_{ik}^{(\lambda)} y_i^{(\lambda)} - \sum_{i=1}^{n} w_{ik}^{(\lambda)} z_k &= 0 \\ \sum_{i=1}^{n} w_{ik}^{(\lambda)} z_k &= \sum_{i=1}^{n} w_{ik}^{(\lambda)} y_i^{(\lambda)} \\ \Longrightarrow \hat{z}_k^{(\lambda)} &= \frac{\sum_{i=1}^{n} w_{ik}^{(\lambda)} y_i^{(\lambda)}}{\sum_{i=1}^{n} w_{ik}^{(\lambda)}} \end{aligned} \tag{2.10.8}$$

Similarly

$$\begin{aligned}
\frac{\partial \ell^*}{\partial \sigma} &= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\left[-\frac{1}{\sigma}+\frac{1}{\sigma^3}(y_i^{(\lambda)}-z_k)^2\right]=0\\
-\sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)} &+ \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{w_{ik}^{(\lambda)}}{\sigma^2}(y_i^{(\lambda)}-z_k)^2=0\\
n &= \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{w_{ik}^{(\lambda)}}{\sigma^2}(y_i^{(\lambda)}-z_k)^2\\
n\sigma^2 &= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik}^{(\lambda)}(y_i^{(\lambda)}-z_k)^2\\
\Longrightarrow \hat{\sigma}^{2(\lambda)} &= \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{w_{ik}^{(\lambda)}(y_i^{(\lambda)}-z_k)^2}{n} \qquad (2.10.9)
\end{aligned}$$

and $\hat{\pi}_k^{(\lambda)}$ is as in equation (2.4.14).

Replacing the results into Equation (2.10.4) we get the non-parametric profile log-likelihood function.

$$\ell_P(\lambda)=\sum_{i=1}^{n}\log\Big[\sum_{k=1}^{K}\hat{\pi}_k\Big(-\frac{1}{2}\log 2\pi-\log\hat{\sigma}^{(\lambda)}-\frac{1}{2\hat{\sigma}^{2(\lambda)}}(y_i^{(\lambda)}-\hat{z}_k^{(\lambda)})^2+(\lambda-1)\log y_i\Big)\Big] \qquad (2.10.10)$$

Now let

$$\begin{aligned}
\xi_{ik}^{(\lambda)} &= y_i^{(\lambda)}-\hat{z}_k^{(\lambda)} \qquad (2.10.11)\\
&= y_i^{(\lambda)}-\frac{\sum_{m=1}^{M} w_{mk}^{(\lambda)}y_m^{(\lambda)}}{\sum_{m=1}^{M} w_{mk}^{(\lambda)}}\\
&= \frac{\sum_{m=1}^{M} w_{mk}^{(\lambda)}y_i^{(\lambda)}-\sum_{m=1}^{M} w_{mk}^{(\lambda)}y_m^{(\lambda)}}{\sum_{m=1}^{M} w_{mk}^{(\lambda)}}\\
\Longrightarrow \xi_{ik}^{(\lambda)} &= \frac{\sum_{m=1}^{M} w_{mk}^{(\lambda)}(y_i^{(\lambda)}-y_m^{(\lambda)})}{\sum_{m=1}^{M} w_{mk}^{(\lambda)}} \qquad (2.10.12)
\end{aligned}$$

The non-parametric profile log-likelihood function is thus

$$\ell_P(\lambda)=\sum_{i=1}^{n}\log\Big[\sum_{k=1}^{K}\hat{\pi}_k\Big(-\frac{1}{2}\log 2\pi-\log\hat{\sigma}^{(\lambda)}-\frac{1}{2\hat{\sigma}^{2(\lambda)}}(\xi_{ik}^{(\lambda)})^2 +(\lambda-1)\log y_i\Big)\Big] \qquad (2.10.13)$$

The non-parametric profile log-likelihood can then be written as

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \Big[\sum_{k=1}^{K} \hat{\pi}_k \hat{f}_{ik}^{(\lambda)} \Big] \tag{2.10.14}$$

where $\hat{f}_{ik}^{(\lambda)} = f(y_i, \lambda | \hat{z}_k)$. The non-parametric profile maximum likelihood (NPPML) is therefore given by

$$\hat{\lambda} = arg \max_{\lambda} \ \ell_P(\lambda) \tag{2.10.15}$$

which can be found through a grid search over λ.

2.11 Applications

Example 2.11.1. the Airline Passenger data

We consider the `AirPassengers` data from the R library **datasets** (R Core Team, 2016) which is a monthly airline passenger numbers from 1949-1960 of size $n = 144$. In this example, we follow the steps given in the flow chart in Figure 2.8.1. First, we search for the optimal `tol` that can be used to set the initial values for a set of K values in order to obtain the best solution.

R Note:

Import the `AirPassengers` data into R, then:

```
library(boxcoxmix)

AirP<-Kfind.boxcox(AirPassengers~1,data=AirPassengers,

find.k = c(2,8),steps.tol =15 ,model.selection = "aic", lambda=1)

#Minimal AIC: 1775.689 at K= 4
```

K	1	2	3	4	5	6	7	8
optim `tol`	–	1.4	1.1	0.4	0.3	1.1	0.7	0.4
$-2\ell_P(\lambda)$	1787.371	1772.195	1762.054	1757.689	1755.062	1752.650	1751.885	1746.646
AIC	1789.371	1782.195	1776.054	1775.689	1777.062	1778.650	1781.885	1780.646
BIC	1792.341	1797.044	1796.843	1802.417	1809.730	1817.258	1826.432	1831.133

Table 2.11.1: Comparison of results from the untransformed `AirPassengers` data ($\lambda = 1$), using K from 1 to 8

Concerning the choice of K, it is apparent from Table 2.11.1 that there is no gain in going up more than $K = 4$ as the AIC values in fact increase when doing so. There is a consistent improvement, however, when increasing the number of mass points from $K = 1$ to $K = 4$. In contrast, BIC seems to favour fixed effect model. In this example, for the untransformed data, the appropriate value of K is 1. By using a range of K values together with the optimal `tol` for each number of classes, we can perform a grid search over λ and then optimize over this grid.

K	1	2	3	4	5	6	7	8
$\hat{\lambda}$	0.1125	0.7250	0.8125	0.6375	0.7250	0.9875	1.1625	0.9875
$-2\ell_P(\hat{\lambda})$	1768.777	1763.530	1758.627	1754.678	1752.177	1752.590	1751.066	1746.580
AIC	1772.777	1771.530	1770.627	1770.678	1772.177	1776.590	1779.066	1778.580
BIC	1778.716	1783.409	1788.446	1794.437	1801.875	1812.228	1820.644	1826.097

Table 2.11.2: Comparison of results from the transformed `AirPassengers` data using K from 1 to 8

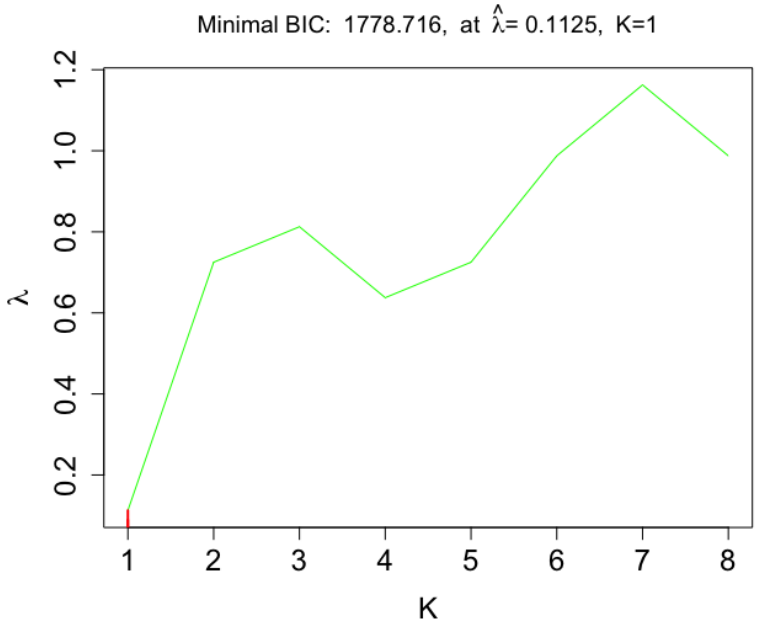


Figure 2.11.1: $\hat{\lambda}$ as a function of K with the optimal `tol` for each K of modelling `AirPassengers` data

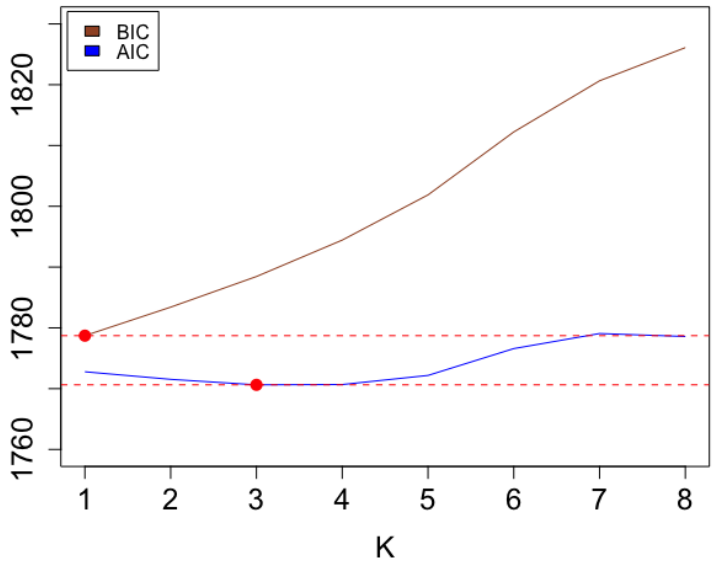


Figure 2.11.2: AIC and BIC values of the model after applying the response transformation to the `AirPassengers` data for $K \in [1, 8]$

The λ/K trade-off is rather clear from Figure 2.11.1 and Table 2.11.2. For larger K no need for any transformation, perhaps as excess variation is already

accounted for. Alternatively, with $K = 1$ something like a log-transformation works. Comparing the `BIC` and disparity values $-2\ell_P(\lambda)$ for the transformed response of the pure mixture models for $K = 1$ with those for $K > 1$, it appears that the smallest BIC value occurs when the fixed effect model is fitted for the transformed case ($\hat{\lambda} = 0.1125$) with BIC = 1778.716, see Table 2.11.2 and Figure 2.11.2.

Example 2.11.2. the Internet Usage data

We consider the `WWWusage` data from the R library **datasets** (R Core Team, 2016) which is a time series of 100 minutes recording how many users an internet server had every minute. The paper by Qarmalah et al. (2018) indicated that the data follows a mixture of either three or four normal distributions. To examine that, as in the previous example, we follow the flow chart in Figure 2.8.1 by searching for the optimal `tol` for each number of classes and then applying the Box–Cox transformation for each number of classes with their optimal `tol`. The model which minimizes either AIC or BIC with a small number of classes is selected as the best–fitting model. The AIC and BIC of the untransformed data ($\lambda = 1$) differ continuously over the specified range of K (from two to eight) as shown in Table 2.11.3, the best model which minimizes the AIC is the 8–component model with AIC = 972.814, whereas the lowest BIC value is 1004.635 at K= 4. In this case, the optimal number of classes is taken as 4.

K	1	2	3	4	5	6	7	8
optim `tol`	–	1.1	0.6	0.2	0.1	0.1	0.2	0.1
$-2\ell_P(\lambda)$	1021.561	1016.7139	992.3199	963.1884	963.1885	958.0025	955.6785	938.8141
AIC	1023.561	1026.7139	1006.3199	981.1884	985.1885	984.0025	985.6785	972.8141
BIC	1026.166	1039.740	1024.556	1004.635	1013.845	1017.870	1024.756	1017.102

Table 2.11.3: Comparison of results from the untransformed `WWWusage` data ($\lambda = 1$), using K from 1 to 8

K	1	2	3	4	5	6	7	8
$\hat{\lambda}$	0.1417	0.1417	1.0167	0.9	0.9	0.6083	1.3667	0.725
$-2\ell_P(\lambda)$	1015.7589	1014.7539	992.5683	963.1301	963.1301	957.7279	953.9402	936.7462
AIC	1019.7589	1022.7539	1004.5683	979.1301	983.1301	981.7279	981.9402	968.7462
BIC	1024.9693	1033.1746	1020.1993	999.9715	1009.1818	1012.9899	1018.4125	1010.4290

Table 2.11.4: Comparison of results from the transformed `WWWusage` data ($\lambda = 1$), using K from 1 to 8

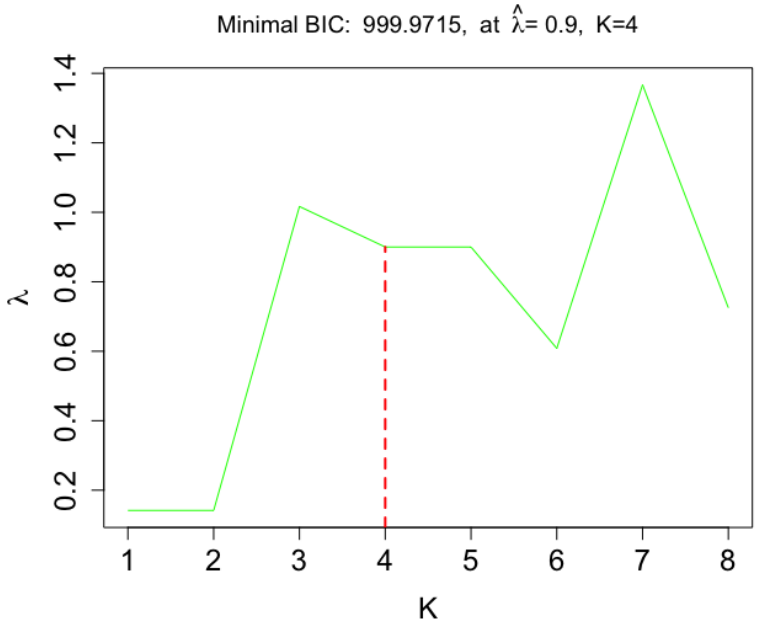


Figure 2.11.3: $\hat{\lambda}$ as a function of K with the optimal `tol` for each number of classes of modelling the `WWWusage` data

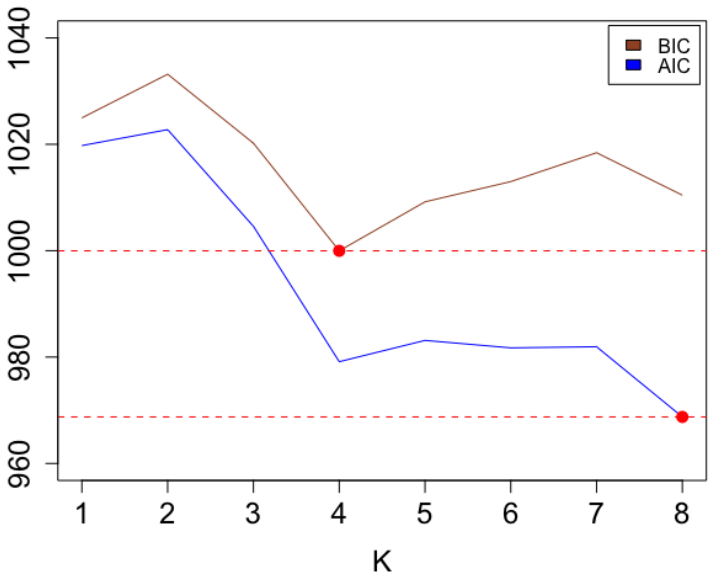


Figure 2.11.4: AIC and BIC values of the model after applying the response transformation to the `WWWusage` data for $K \in [1, 8]$

Comparing the results of both criteria for the transformed data for each K in Figure 2.11.4 and Table 2.11.4, we notice that the minimal AIC value occurred at $K = 8$ with $\hat{\lambda}$ =0.725 (AIC=968.7462) while the minimal BIC value occurred at $K = 4$ with $\hat{\lambda} = 0.9$ (BIC=999.9715). More evidence of the λ/K trade-off is shown in Table 2.11.4. Again, the four–component model is considered as the best model. Figure 2.11.3 shows that the best estimate of λ that maximizes the non–parametric profile log–likelihood is close to the value of 1, suggesting that no transformation is needed. That supports the suggestion given in paper by Qarmalah et al. (2018) that the `WWWusage` data follows a normal distribution subject to heterogeneity. The residuals plots for `WWWusage` data before and after applying the response transformation for $K \in [1, 4]$ are shown in Appendix A.

2.12 Discussion

It is common to normalize the non–normal data via a normalizing transformation prior to analysis. In order to select an appropriate transformation parameter for the linear model with random effects of unspecified distribution we have developed methodology for simultaneous response transformation and estimation of regression parameters. This is achieved by extending the "Nonparametric Maximum Likelihood" towards a "Nonparametric Profile Maximum Likelihood" technique.

In this Chapter, we have introduced a new R package **boxcoxmix** that identifies the appropriate power transformation for achieving normality of the response distribution in random effect models with a non–parametric setting. To the best of our knowledge, there is no other widely available statistical package that has implemented the Box–Cox power transformation of the linear mixed effects model with an unspecified random effect distribution. **boxcoxmix** is able to estimate the transformation and regression parameters simultaneously through its main function `optim.boxcox()` but K has to be fixed in this process. This function operates similarly to the existing R function `boxcox()`, by creating a profile likelihood and carrying out a grid search over the transformation parameter λ but our method is based on non–parametric estimation of λ. It is noted that, just as in `boxcox()`, this procedure cannot make use of built–in R optimization routines such as `optim()` or `optimize()` since the profile likelihood itself depends on estimated parameters, estimation of which involves a full EM algorithm. In addition, **boxcoxmix** also can be used to fit models with fixed value of λ using function `np.boxcoxmix()`, and to perform a grid search over `tol` using the function `tolfind.boxcox()` to identify

optimal starting values for the mass points. Our package provides some further diagnostic tools, such as a QQ–plot and a control chart of residuals, which help validating the need for transformation.

To assess the performance of the proposed approach, we conducted two simulation studies. In the first simulation study in which the residuals of the fitted model were normal on the original scale before being non–normal by applying the inverse transformation in the simulation process, we have seen that the method is able to transform the data back to its original position when λ's are fixed. Also, the simulations where λ's are unknown showed that the method is able to spot the true value of λ. The results demonstrated the effectiveness of the proposed method in estimating the regression parameters. Furthermore, comparing the robust estimate of the standard deviation with the EM–based standard errors of the regression parameter estimates revealed the biases of the estimated standard errors of the parameters due to ignoring the variation of the estimates of the regression parameters that results from the variables with the transformation parameters estimates. The second study based on non–normal distribution of the fitted model on the original scale due to the effect of the discrete distribution for the random effects and the distributions of the covariates on the underlying distribution of the residuals, which was determined by the graphical method for normality given in Appendix A. The related results showed a large bias when $\lambda \neq 0$, the cause of which was considered that the generated data had originally a non–normal distribution prior to applying the inverse of the transformation. Thus, in order to tranform the data to a close-to-normal distribution our method selected $\hat{\lambda}$ that transforms the data far from its original scale. Taken together, the simulation results indicate that there is a strong relationship

between the regression and transformation parameters. The bias in the regression parameter estimates increases as the bias in the transformation parameter estimates increases. In the Gaussian random effect distribution case, Gurka (2004) showed the correlation between the estimation of the transformation and regression parameters. He observed that the bias in the estimate of the transformation parameter results in incorrect conclusions about the estimation of the regression parameters. He concluded that transforming the response impacts the inference about the fixed effect when compared to the response with no transformation. Furthermore, our simulation results suggest that the estimation method may be influenced by the varying structures of the simulated dataset.

The simulation results also showed a strong consistency of the parameter estimates when the log–transformation is the most appropriate transformation for the simulated data. However, as demonstrated by Gurka (2004) in the Gaussian random effect distribution case, there may be a computational problem when $\lambda = 0$ is the most suitable transformation. In the univariate case, Asar et al. (2017) proposed different approaches to estimate the Box-Cox power transformation parameter and implement simulation studies to compare their effectiveness, and the related results indicated that all of the methods, including the one that was not preferred to estimate λ, performed well at $\lambda = 0$ regardless of what design is used to generate the data. Changyong et al. (2014) showed that the log transformation does not necessarily make data conform more closely to the normal distribution. From this arises the question whether the restriction on the response to be greater than zero has an effect on the results of the log–transformation. Accordingly, it would be interesting to examine the possibility of using a small value of λ that is close to

zero for transformaing the response instead of log–transformation when $\lambda = 0$ is selected as the optimum. This solution has been taken before; see, for instance, Gurka (2004). In further research, the effect of the sample size of the generated data in the estimation results of the transformation should be studied.

Additionally, we have shown how **boxcoxmix** can successfully fit models through response transformation rather than adjustment of the response distribution. The examples have demonstrated that the proposed approach works well in finding the model with maximum likelihood. As in the univariate case, the Box-Cox transformation does not guarantee that the assumptions of homoscedasticity and normality of the response distribution in the random effects model is met after applying the transformation, however, it provides a data for which the homoscedasticity and normality assumptions are more reasonable than not applying the transformation at all. All transformed models using $\hat{\lambda}$ that were obtained by the **boxcoxmix** function `optim.boxcox()` gave substantially better fits than the untransformed models, when considering the AIC and BIC criteria or the disparity ($-2\ell_P(\lambda)$). It should be added that it is not possible to report a simple likelihood–based confidence interval for $\hat{\lambda}$ as in R function `boxcox()`, the reason being that the likelihood in the considered model class is highly non–concave, as visible for instance from Figure 2.9.2. Hence, when faced with the decision on whether or not needing to transform the response, not only the value of $\hat{\lambda}$ but also the relevant model selection criteria such as AIC and BIC should be taken into account. It is then essential that these are always based on likelihoods which are reported on the original response scale, as in model (2.4.5), of course, this is the case for the values $-2\ell_P(\lambda)$, AIC and BIC provided in our summary output. In contrast, comparing the coefficients make no sense since

the estimates of β vary greatly with a very minor change of choice of $\hat{\lambda}$.

In the Example 2.11.2, $\hat{\lambda}$ for classical `boxcox` was much further away from $\lambda = 1$ than for `boxcoxmix`, therefore, it is beneficial to test the need for a transformation of the response of random effect model even if the classical `boxcox` does need transformation! This gives us further support for our method because it can tell us if the data really needs to be transformed or only the right number of components needs to be found in order to have a normal distribution. Moreover, Example 2.9.2 showed a strong need of a transformation as we increase the number of classes (Figure 2.9.4). Concerning the choice of the number of components, Lukociene (2010) indicated that the NPML estimation may yield an unnecessarily high number of components. McLachlan and Peel (2004) and Aitkin et al. (2005) suggested using the penalized-log–likelihood criteria, such as AIC and BIC, by increasing the number of components in the fitted model until the decrease in these criteria stabilizes. As demonstrated by the real data examples, a large number of classes is required to minimize the disparity although well-fitting models can be found with a smaller number of components. The experimental results verify the accuracy and the efficiency of the proposed approach and its implemented package **boxcoxmix**.

Chapter 3

Box-Cox transformations for two–level models

3.1 Introduction

In the previous chapter, we have presented a general introduction to the Box–Cox transformation for the univariate linear model. Then, we have explored the random effect model that accounts for an individual random effect on an observation. Finally, the transformation has been extended to the random effect models. A brief description of the **boxcoxmix** package with real data examples and simulations have been presented and discussed. However, assume one wishes to analyze a dataset containing observations that share a random effect (e.g. classes or schools), and repeated individual observations over time (longitudinal data). This leads to the two–level variance component models, which we wish to introduce in this Chapter, and onto which we will apply the Box–Cox transformation in a similar manner as

for the random effect models with some associated changes.

Section 3.2 provides an introduction to the two–level variance component models, followed by a real data example. In Section 3.3, the Box-Cox transformation is extended to the variance component model, along with some software descriptions in Section 3.4. We demonstrate the applicability of the proposed approach using simulated and real data examples in Sections 3.5 and 3.6. Finally, we provide a discussion in Section 3.7.

3.2 Two-level models

For data with a two–level structure, such as longitudinal data, correlation of responses within upper–level units can be induced by adding a random effect z_i to the linear predictor $x_{ij}^T\beta$, with the upper-level indexed by $i = 1, \ldots, r$, and the lower-level indexed by $j = 1, \ldots, n_i$, $\sum_i^r n_i = n$. Conditional on the random effect, the responses y_{ij} are independently distributed with mean function

$$E(y_{ij}|z_i) = x_{ij}^T\beta + z_i, \tag{3.2.1}$$

which is also known as a variance component model. As in Chapter 2, we make no assumption about the distribution of the z_i. When $n_i \equiv 1$, it reduces to the random effect models, presented in Chapter 2.

3.2.1 Estimation of finite mixtures

This is a simple variant of estimation in Chapter 2, with the same issues of iteration.

The conditional probability density function of y_{ij} given z_i is given by

$$f(y_{ij}|z_i) = \phi(y_{ij}; x_{ij}^T\beta + z_i, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_{ij} - x_{ij}^T\beta - z_i)^2\right] \tag{3.2.2}$$

As in Chapter 2, the likelihood is again approximated using NPML estimation (Aitkin et al., 2009).

$$L(\beta, \sigma^2, g) = \prod_{i=1}^{r} \int \left[\prod_{j=1}^{n_i} f(y_{ij}|z_i)\right] g(z_i)dz_i \approx \prod_{i=1}^{r} \sum_{k=1}^{K} \pi_k m_{ik} \tag{3.2.3}$$

where $m_{ik} = \prod_{j=1}^{n_i} f(y_{ij}|z_k)$. The log-likelihood is then

$$\ell = \log L = \log\Big(\prod_{i=1}^{r} \sum_{k=1}^{K} \pi_k m_{ik}\Big) = \sum_{i=1}^{r} \log\Big(\sum_{k=1}^{K} \pi_k m_{ik}\Big) \tag{3.2.4}$$

Using notation as defined in (2.3.7), the "complete data" likelihood would be

$$L^* = \prod_{i=1}^{r} \prod_{k=1}^{K} (\pi_k m_{ik})^{G_{ik}} \tag{3.2.5}$$

The complete log-likelihood is thus

$$\ell^* = \log L^* = \sum_{i=1}^{r} \sum_{k=1}^{K} [G_{ik} \log \pi_k + G_{ik} \log m_{ik}] \tag{3.2.6}$$

where

$$\begin{aligned}
\log m_{ik} &= \sum_{j=1}^{n_i} \log f(y_{ij}|z_k) \\
&= \sum_{j=1}^{n_i} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_{ij} - x_{ij}^T\beta - z_k)^2\right]\right) \\
&= \sum_{j=1}^{n_i} \left(-\frac{1}{2}\log 2\pi - \log\sigma - \frac{1}{2\sigma^2}(y_{ij} - x_{ij}^T\beta - z_k)^2\right) \\
&= -\frac{n_i}{2}\log 2\pi - n_i \log\sigma - \frac{1}{2\sigma^2}\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2,
\end{aligned} \tag{3.2.7}$$

then

$$\ell^* = \sum_{i=1}^{r}\sum_{k=1}^{K}\left[G_{ik}\log\pi_k + G_{ik}\left(-\frac{n_i}{2}\log 2\pi - n_i\log\sigma - \frac{1}{2\sigma^2}\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2\right)\right]. \tag{3.2.8}$$

We apply the EM approach as before, with the following adjustments:

E-step: This is similar to that in (2.3.12), but with f_{ik} replaced by m_{ik}.

M-step: Calculate $\hat{z}_k, \hat{\sigma}^2, \hat{\beta}$ and $\hat{\pi}_k$ using the current w_{ik} that defined in the E-step,

$$\begin{aligned}
\frac{\partial\ell^*}{\partial z_k} = -\frac{1}{2\sigma^2}\sum_{i=1}^{r} 2\,w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)\right](-1) &= 0\\
\sum_{i=1}^{r} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)\right] &= 0\\
\sum_{i=1}^{r}\sum_{j=1}^{n_i} w_{ik}z_k = \sum_{i=1}^{r} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta)\right]&\\
\sum_{i=1}^{r} n_i w_{ik}z_k = \sum_{i=1}^{r} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta)\right]&\\
\Longrightarrow \hat{z}_k = \frac{\sum_{i=1}^{r} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta)\right]}{\sum_{i=1}^{r} n_i w_{ik}}&
\end{aligned} \tag{3.2.9}$$

Similarly

$$\begin{aligned}
\frac{\partial\ell^*}{\partial\beta} = -\frac{1}{2\sigma^2}\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} 2\,w_{ik}(-x_{ij})(y_{ij} - x_{ij}^T\beta - z_k) &= 0\\
\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}x_{ij}(y_{ij} - x_{ij}^T\beta - z_k) &= 0\\
\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}x_{ij}y_{ij} - \sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}x_{ij}x_{ij}^T\beta - \sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}x_{ij}z_k &= 0\\
\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}y_{ij}\sum_{k=1}^{K} w_{ik} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}x_{ij}^T\beta\sum_{k=1}^{K} w_{ik} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}z_k &= 0
\end{aligned}$$

$$\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}y_{ij} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}x_{ij}^T\beta - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}z_k = 0$$

$$\begin{aligned}\implies \hat{\beta} &= \left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}x_{ij}^T\right)^{-1}\left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}y_{ij} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}z_k\right)\\ &= \left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}x_{ij}^T\right)^{-1}\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\left(y_{ij} - \sum_{k=1}^{K} w_{ik}z_k\right) \qquad (3.2.10)\end{aligned}$$

Equation (3.2.10) in matrix notation is

$$\hat{\beta} = \left(\underbrace{\underbrace{X^T}_{\mathrm{p}\times n}\quad\underbrace{X}_{n\times \mathrm{p}}}_{\mathrm{p}\times\mathrm{p}}\right)^{-1}\underbrace{X^T}_{\mathrm{p}\times n}\left(\underbrace{\underbrace{Y}_{n\times 1} - \underbrace{\underbrace{W}_{n\times \mathrm{K}}\quad\underbrace{Z}_{\mathrm{K}\times 1}}_{n\times 1}}_{n\times 1}\right) \qquad (3.2.11)$$

where

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_{n1} \\ y_1 \\ \vdots \\ y_{n2} \\ \vdots \\ \vdots \\ y_1 \\ \vdots \\ y_{nr} \end{pmatrix}\begin{matrix}\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_1\\ \left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_2\\ \\ \\ \\ \left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_r\end{matrix}, \qquad X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_1 1} & x_{n_1 2} & \cdots & x_{n_1 p} \\ x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_2 1} & x_{n_2 2} & \cdots & x_{n_2 p} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_r 1} & x_{n_r 2} & \cdots & x_{n_r p} \end{pmatrix}\begin{matrix}\left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_1\\ \left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_2\\ \\ \\ \\ \left.\begin{matrix}\\ \\ \\ \end{matrix}\right\} n_r\end{matrix}$$

$$W = \left.\begin{pmatrix} w_{11} & \cdots & \cdots & w_{1K} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ w_{n_r1} & \cdots & \cdots & w_{n_rK} \end{pmatrix}\right\} n \text{ times} \qquad \text{and} \qquad Z = \begin{pmatrix} z_1 \\ \vdots \\ \vdots \\ z_K \end{pmatrix}$$

and the score for σ is

$$\begin{aligned} \frac{\partial \ell^*}{\partial \sigma} &= \sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}\left(-\frac{n_i}{\sigma} + \frac{1}{\sigma^3}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2\right]\right) = 0 \\ \Longrightarrow \sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik} n_i \sigma^2 &= \sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2\right] \\ \Longrightarrow \hat{\sigma}^2 &= \frac{\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2\right]}{\sum_{i=1}^{r} n_i \sum_{k=1}^{K} w_{ik}} \\ \Longrightarrow \hat{\sigma}^2 &= \frac{\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}\left[\sum_{j=1}^{n_i}(y_{ij} - x_{ij}^T\beta - z_k)^2\right]}{\sum_{i=1}^{r} n_i} \end{aligned} \tag{3.2.12}$$

The derivatives of the log-likelihood with respect to π_k as in Section 2.3, but with n replaced by r.

$$\hat{\pi}_k = \frac{\sum_{i=1}^{r} w_{ik}}{r}. \tag{3.2.13}$$

3.2.2 Existing R implementation: `allvc()`

To fit variance component models, we can use the **npmlreg** function `allvc()` (Einbeck et al., 2014). Similar to the case in the function `alldist()` described in Section 2.3.2, the function `allvc()` relies on the output of the function `glm()` rather than computing (3.2.10),(3.2.9),(3.2.12) and (3.2.13) directly.

Example 3.2.1. the heights of boys in Oxford data

The dataset has been analyzed by Aitkin et al. (2009). The heights of 26 boys in Oxford were recorded on nine equally spaced occasions over two years, yielding a total of 234 observations (**nlme**; Pinheiro et al., 2016). The response variable height is defined as the `height` of the boy in (`cm`), associated with the covariate `age` that is the standardized age (`dimensionless`). Actual heights attained at each age are shown in Figure 3.2.1 for each boy. The individual boys are represented by points joined by lines.

R Note:

Import the `Oxboys` data into R, then:

```
Oxboys$boy <- gl(26,9)
Oxboys$tage <- Oxboys$age+13

plot(Oxboys$age[Oxboys$boy==1],Oxboys$height[Oxboys$boy==1],
ylim=c(125,175),type='b',pch=1,xlab='age',ylab='height')

for (i in 2:nlevels(Oxboys$Subject))
  lines(Oxboys$age[Oxboys$boy==i],Oxboys$height[Oxboys$boy==i],
        pch=1,col=i,type='b')
```

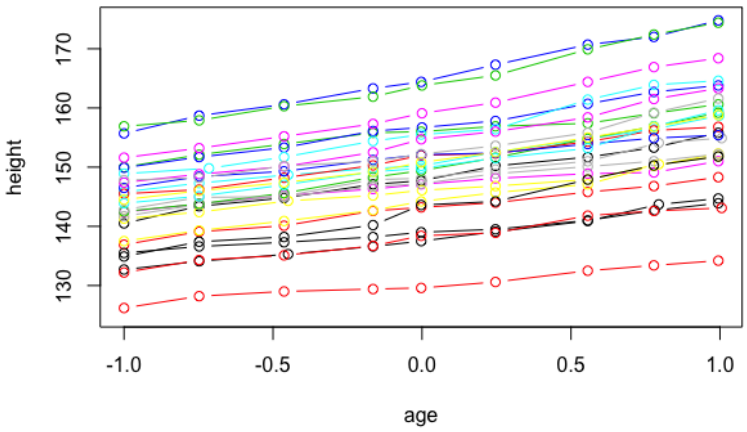


Figure 3.2.1: Heights of 26 boys in Oxford over two years.

The function `allvc()` is used to fit the variance component model,

$$E(y_{ij}|z_i) = \texttt{age}_j + z_i \tag{3.2.14}$$

where z_i is a boy–specific random effect and $\texttt{age}_j$ is the j-th standardized age measurement, $j = 1, \ldots, 9$, which is equal for all boys for fixed j.

R Note:

```
library(npmlreg)
 Oxboys.vc <- allvc(height~age,random=~1|boy,data=Oxboys,
    random.distribution="np",k=8)
summary(Oxboys.vc)
 # Call:  allvc(formula = height ~ age, random = ~1 | boy,
  data = Oxboys, k = 8, random.distribution = "np")
 #
```

```
# Coefficients:
#         Estimate Std. Error   t value
# age     6.523808 0.05599468  116.5076
# MASS1 130.200172 0.18461208  705.2635
# MASS2 138.416626 0.10659164 1298.5692
# MASS3 143.382396 0.10659052 1345.1702
# MASS4 147.350112 0.08256950 1784.5585
# MASS5 151.267275 0.06978777 2167.5326
# MASS6 155.789087 0.09230962 1687.6798
# MASS7 159.521547 0.18461217  864.0901
# MASS8 164.883638 0.13054343 1263.0559
#
# Mixture proportions:
#   MASS1       MASS2       MASS3       MASS4       MASS5
# 0.03846154  0.11538462  0.11538469  0.19230765  0.26921962
# MASS6       MASS7       MASS8
# 0.15385725  0.03846155  0.07692308
#
# Component distribution - MLE of sigma:    1.433
# Random effect distribution - standard deviation:    7.917343
#
# -2 log L:     931.4     Convergence at iteration  10
```

```
plot(Oxboys.vc)
```

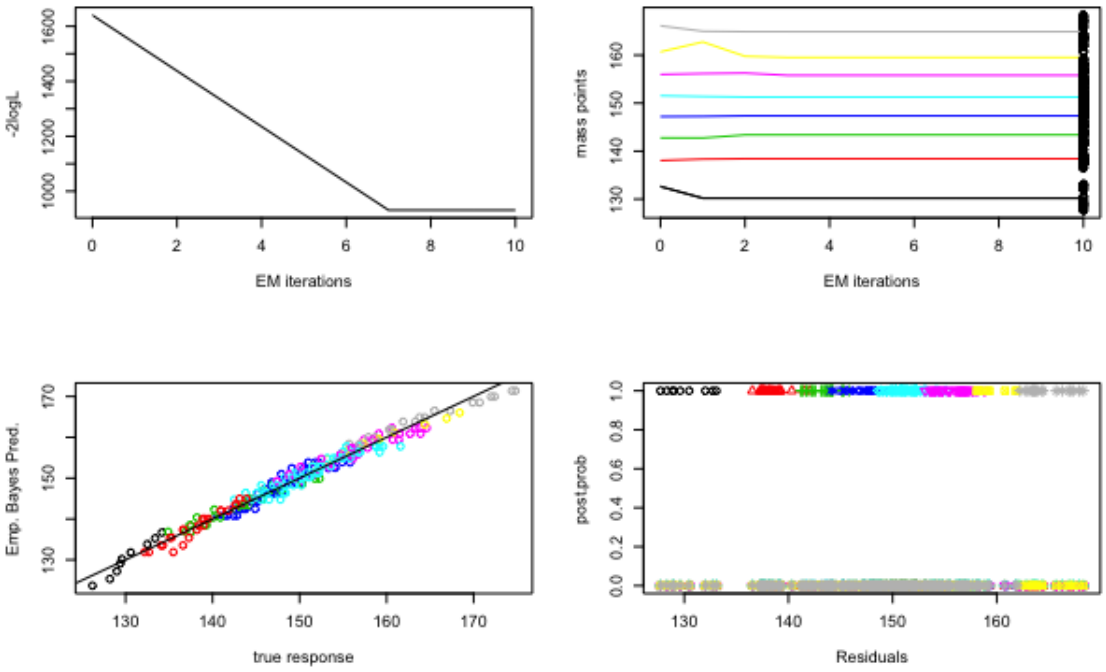


Figure 3.2.2: Fitting the variance component with NPML to the `Oxboys` data using the function `allvc`, with `k=6` and `tol=0.5`

Figure 3.2.2 shows how the posterior splits the data into 8 distinct classes.

3.3 Box-Cox transformations for two-level models

In this Section, we shall see how the approach for applying the Box–Cox transformation to this model closely parallels that for random effect model but the EM algorithm gets a bit more complicated, yields a quite straightforward way to extend the transformation to the two-level models. Under the scenario of model (3.2.1), the

transformation by Box and Cox (1964) can be written as

$$y_{ij}^{(\lambda)} = \begin{cases} \dfrac{y_{ij}^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_{ij} & (\lambda = 0) \end{cases} \tag{3.3.1}$$

and that for $y_{ij} > 0$, $i = 1, ..., r$, $j = 1, ..., n_i$, and $\sum n_i = n$. From the inversion of 3.3.1 we get

$$\hat{y}_{ij} = \begin{cases} \left(1 + \lambda\eta_{ij}\right)^{1/\lambda} & (\lambda \neq 0), \\ e^{\eta_{ij}} & (\lambda = 0) \end{cases} \tag{3.3.2}$$

where $\eta_{ij} = x_{ij}^T\beta + z_i$.

3.3.1 Estimation of finite mixtures

In the case of two-level variance component models, it is assumed that there is a value of λ for which,

$$y_{ij}^{(\lambda)}|z_i \sim N(x_{ij}^T\beta + z_i, \sigma^2) \tag{3.3.3}$$

where z_i again has an unknown mixing distribution $g(z_i)$. Taking account of the Jacobian of the transformation from y_{ij} to $y_{ij}^{(\lambda)}$, the conditional probability density function of y_{ij} given z_i is given by

$$f(y_{ij}, \lambda|z_i) = \phi(y_{ij}^{(\lambda)}; x_{ij}^T\beta + z_i, \sigma^2)y_{ij}^{\lambda-1} = \frac{y_{ij}^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_i)^2\right] \tag{3.3.4}$$

The likelihood can now be approximated using NPML estimation (Aitkin et al., 2009).

$$L(\lambda, \beta, \sigma^2, g) = \prod_{i=1}^{r} \int \left[\prod_{j=1}^{n_i} f(y_{ij}, \lambda|z_i)\right] g(z_i)dz_i \approx \prod_{i=1}^{r} \sum_{k=1}^{K} \pi_k m_{ik}^{(\lambda)} \tag{3.3.5}$$

where $m_{ik}^{(\lambda)} = f(y_{ij}, \lambda | z_k)$. The log-likelihood is then

$$\ell = \log L = \log \Big(\prod_{i=1}^{r} \sum_{k=1}^{K} \pi_k m_{ik}^{(\lambda)} \Big) = \sum_{i=1}^{r} \log \Big(\sum_{k=1}^{K} \pi_k m_{ik}^{(\lambda)} \Big) \tag{3.3.6}$$

Refer to (2.3.7), the "complete data" log-likelihood would be

$$\ell^* = \log L^* = \sum_{i=1}^{r} \sum_{k=1}^{K} \Big[G_{ik} \log \pi_k + G_{ik} \log m_{ik}^{(\lambda)} \Big] \tag{3.3.7}$$

where

$$\begin{aligned}
\log m_{ik}^{(\lambda)} &= \sum_{j=1}^{n_i} \log f(y_{ij}, \lambda | z_k) \\
&= \sum_{j=1}^{n_i} \log \left(\frac{y_{ij}^{\lambda-1}}{\sqrt{2\pi\sigma^2}} \exp \Big[-\frac{1}{2\sigma^2} (y_{ij}^{(\lambda)} - x_{ij}^T \beta - z_k)^2 \Big] \right) \\
&= \sum_{j=1}^{n_i} \Big(-\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (y_{ij}^{(\lambda)} - x_{ij}^T \beta - z_k)^2 + (\lambda - 1) \log y_{ij} \Big) \\
&= -\frac{n_i}{2} \log 2\pi - n_i \log \sigma - \frac{1}{2\sigma^2} \sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T \beta - z_k)^2 + (\lambda - 1) \sum_{j=1}^{n_i} \log y_{ij},
\end{aligned} \tag{3.3.8}$$

then

$$\begin{aligned}
\ell^* = \sum_{i=1}^{r} \sum_{k=1}^{K} \Bigg[G_{ik} \log \pi_k + G_{ik} \Bigg(&- \frac{n_i}{2} \log 2\pi - n_i \log \sigma - \frac{1}{2\sigma^2} \sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T \beta - z_k)^2 \\
&+ (\lambda - 1) \sum_{j=1}^{n_i} \log y_{ij} \Bigg) \Bigg].
\end{aligned} \tag{3.3.9}$$

Applying the EM approach to approximate the MLE of the model parameters:

E-step: This is exactly that in (2.4.10), but with $f_{ik}^{(\lambda)}$ replaced by $m_{ik}^{(\lambda)}$.

M-step: Calculate $\hat{z}_k^{(\lambda)}, \hat{\sigma}^{2(\lambda)}, \hat{\beta}^{(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ using current $w_{ik}^{(\lambda)}$,

$$\frac{\partial \ell^*}{\partial z_k} = -\frac{1}{2\sigma^2}\sum_{i=1}^{r} 2\, w_{ik}^{(\lambda)}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)\right](-1) = 0$$

$$\sum_{i=1}^{r} w_{ik}^{(\lambda)}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)\right] = 0$$

$$\sum_{i=1}^{r}\sum_{j=1}^{n_i} w_{ik}^{(\lambda)} z_k = \sum_{i=1}^{r} w_{ik}^{(\lambda)}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta)\right]$$

$$\sum_{i=1}^{r} n_i w_{ik}^{(\lambda)} z_k = \sum_{i=1}^{r} w_{ik}^{(\lambda)}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta)\right]$$

$$\Longrightarrow z_k^{(\lambda)} = \frac{\sum_{i=1}^{r} w_{ik}^{(\lambda)}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta)\right]}{\sum_{i=1}^{r} n_i w_{ik}^{(\lambda)}} \tag{3.3.10}$$

Similarly

$$\frac{\partial \ell^*}{\partial \beta} = -\frac{1}{2\sigma^2}\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} 2\, w_{ik}^{(\lambda)}(-x_{ij})(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k) = 0$$

$$\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}^{(\lambda)} x_{ij}(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k) = 0$$

$$\sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}^{(\lambda)} x_{ij} y_{ij}^{(\lambda)} - \sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}^{(\lambda)} x_{ij} x_{ij}^T\beta - \sum_{i=1}^{r}\sum_{k=1}^{K}\sum_{j=1}^{n_i} w_{ik}^{(\lambda)} x_{ij} z_k = 0$$

$$\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} y_{ij}^{(\lambda)}\sum_{k=1}^{K} w_{ik}^{(\lambda)} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} x_{ij}^T\beta\sum_{k=1}^{K} w_{ik}^{(\lambda)} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k = 0$$

$$\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} y_{ij}^{(\lambda)} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} x_{ij}^T\beta - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k = 0$$

$$\Longrightarrow \beta^{(\lambda)} = \left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} x_{ij}^T\right)^{-1}\left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} y_{ij}^{(\lambda)} - \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k\right)$$

$$= \left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij} x_{ij}^T\right)^{-1}\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\left(y_{ij}^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)} z_k\right) \tag{3.3.11}$$

and

$$\frac{\partial \ell^*}{\partial \sigma} = \sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)}\left(-\frac{n_i}{\sigma} + \frac{1}{\sigma^3}\left[\sum_{j=1}^{n_i}(y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)^2\right]\right) = 0$$

$$\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)} n_i \sigma^2 = \sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)^2\right]$$

$$\implies \sigma^{2(\lambda)} = \frac{\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)^2\right]}{\sum_{i=1}^{r} n_i \sum_{k=1}^{K} w_{ik}^{(\lambda)}}$$

$$= \frac{\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T\beta - z_k)^2\right]}{\sum_{i=1}^{r} n_i} \tag{3.3.12}$$

and the average posterior probability $\pi_k^{(\lambda)}$ for component k is as in (3.2.13), but with w_{ik} replaced by $w_{ik}^{(\lambda)}$.

This leads to the four reconciled equations (emphasizes the dependence on λ explicitly)

$$\hat{z}_k^{(\lambda)} = \frac{\sum_{i=1}^{r} w_{ik}^{(\lambda)} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T\hat{\beta}^{(\lambda)})\right]}{\sum_{i=1}^{r} n_i w_{ik}} \tag{3.3.13}$$

$$\hat{\beta}^{(\lambda)} = \left(\sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}x_{ij}^T\right)^{-1} \sum_{i=1}^{r}\sum_{j=1}^{n_i} x_{ij}\left(y_{ij}^{(\lambda)} - \sum_{k=1}^{K} w_{ik}^{(\lambda)}\hat{z}_k^{(\lambda)}\right) \tag{3.3.14}$$

$$\hat{\sigma}^{2(\lambda)} = \frac{\sum_{i=1}^{r}\sum_{k=1}^{K} w_{ik}^{(\lambda)} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T\hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)})^2\right]}{\sum_{i=1}^{r} n_i} \tag{3.3.15}$$

$$\hat{\pi}_k^{(\lambda)} = \frac{\sum_{i=1}^{r} w_{ik}^{(\lambda)}}{r} \tag{3.3.16}$$

Equation (3.3.14) in matrix notation is

$$\hat{\beta}^{(\lambda)} = \left(\underbrace{\underbrace{X^T}_{\mathrm{p}\times n}\quad\underbrace{X}_{n\times \mathrm{p}}}_{\mathrm{p}\times\mathrm{p}}\right)^{-1} \underbrace{X^T}_{\mathrm{p}\times n} \left(\underbrace{\underbrace{Y^{(\lambda)}}_{n\times 1} - \underbrace{W^{(\lambda)}}_{n\times \mathrm{K}}\quad\underbrace{\hat{Z}^{(\lambda)}}_{\mathrm{K}\times 1}}_{n\times 1}\right)_{n\times 1} \tag{3.3.17}$$

Replacing the results into Equation (3.3.6) we get the non-parametric profile log-likelihood function.

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \left(-\frac{n_i}{2} \log 2\pi - n_i \log \hat{\sigma}^{(\lambda)} \right.\right.$$
$$\left.\left. - \frac{1}{2\hat{\sigma}^{2(\lambda)}} \left[\sum_{j=1}^{n_i} (y_{ij}^{(\lambda)} - x_{ij}^T \hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)})^2 \right] + (\lambda - 1) \log \sum_{j=1}^{n_i} y_{ij} \right) \right] \quad (3.3.18)$$

Now let

$$\xi_{ijk}^{(\lambda)} = y_{ij}^{(\lambda)} - x_{ij}^T \hat{\beta}^{(\lambda)} - \hat{z}_k^{(\lambda)} \quad (3.3.19)$$
$$= y_{ij}^{(\lambda)} - x_{ij}^T \hat{\beta}^{(\lambda)} - \frac{\sum_{m=1}^{r} w_{mk}^{(\lambda)} (y_{mj}^{(\lambda)} - x_{mj}^T \hat{\beta}^{(\lambda)})}{\sum_{m=1}^{r} w_{mk}^{(\lambda)}}$$
$$= \frac{\sum_{m=1}^{r} w_{mk}^{(\lambda)} y_i^{(\lambda)} - \sum_{m=1}^{r} w_{mk}^{(\lambda)} x_i^T \hat{\beta}^{(\lambda)} - \sum_{m=1}^{r} w_{mk}^{(\lambda)} y_m^{(\lambda)} - \sum_{m=1}^{r} w_{mk}^{(\lambda)} x_m^T \hat{\beta}^{(\lambda)}}{\sum_{m=1}^{r} w_{mk}^{(\lambda)}}$$
$$\implies \xi_{ijk}^{(\lambda)} = \frac{\sum_{m=1}^{r} w_{mk}^{(\lambda)} \left((y_{ij}^{(\lambda)} - y_{mj}^{(\lambda)}) - (x_{ij}^T - x_{mj}^T) \hat{\beta}^{(\lambda)} \right)}{\sum_{m=1}^{r} w_{mk}^{(\lambda)}} \quad (3.3.20)$$

The non-parametric profile log-likelihood function is thus

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \left(-\frac{n_i}{2} \log 2\pi - n_i \log \hat{\sigma}^{(\lambda)} - \frac{1}{2\hat{\sigma}^{2(\lambda)}} \sum_{j=1}^{n_i} \left(\xi_{ijk}^{(\lambda)} \right)^2 \right.\right.$$
$$\left.\left. + (\lambda - 1) \log \sum_{j=1}^{n_i} y_{ij} \right) \right] \quad (3.3.21)$$

The non-parametric profile log-likelihood can then be written as

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \hat{m}_{ik}^{(\lambda)} \right]. \quad (3.3.22)$$

where $\hat{m}_{ik}^{(\lambda)} = f(y_{ij}, \lambda | \hat{z}_k)$. The non-parametric profile maximum likelihood (NPPML) estimate of λ is therefore given by

$$\hat{\lambda} = arg \max_{\lambda} \ell_P(\lambda). \quad (3.3.23)$$

which can be found through a grid search over λ.

3.4 Software description

As with the random effect models, the methodology is implemented in the R package **boxcoxmix**. This package is described in detail in Section 2.6. We can use the same functions used for random effect case and in the same way, however, the only difference is that the `groups` argument in the **boxcoxmix** functions was equal 1 (i.e. $n_i \equiv 1$) in the random effect models and here we use the grouping variable instead.

3.5 Simulation study

The simulation study for the Box–Cox transformed variance component model parallels exactly that in Chapter 2. We conduct two scenarios to assess the performance of our approach using fixed and unknown values of the transformation parameters (λ) to estimate the model parameters (β).

We are interested in examining the method's ability to estimate the true parameter values. Therefore, we first simulate data by applying the Box–Cox transformation 'backwards' to a dataset that follows a normal distribution using a set of λ values. Specifically, for each of four given values λ_ℓ, $\ell = 1, 2, 3, 4$, we generate 1000 datasets with 100 observations as follows,

$$\zeta_{ij\ell} = \hat{y}(\eta_{ij}, \lambda_\ell), \;\; i = 1, ..., 20, \; j = 1, ..., 5 \tag{3.5.1}$$

$$\hat{y}(\eta_{ij}, \lambda_\ell) = \begin{cases} \left(1 + \lambda_\ell \eta_{ij}\right)^{1/\lambda_\ell} & (\lambda_\ell \neq 0), \\ e^{\eta_{ij}} & (\lambda_\ell = 0) \end{cases}$$

$$\eta_{ij} = 3\, x_{ij} + z_i + \varepsilon_{ij}$$

$$X \sim U(-4, 4), \qquad \varepsilon \sim N(0, 0.5^2)$$

$$\lambda_1 = 0,\ \ \lambda_2 = 0.5,\ \ \lambda_3 = 1,\ \ \lambda_4 = 2$$

$$z_i \sim \text{Multinomial}\{1, (z_1, \ldots, z_4)|\pi_1, \ldots, \pi_4\}$$

$$z_k = (15, 20, 30, 35) \text{ with masses } \pi_k = 1/4,\ k = 1, ..., 4.$$

Note that $\hat{y}(\cdot)$ denotes the 'backward' Box–Cox–transformation, and that the generated data possess a variance component structure due to the random effect terms z_i. The reader refer to Appendix A to see how data were generated in R.

The simulation process and the estimation method are exactly that for random effect model, discussed in Chapter 2. As before we consider this scenario to test the machinery's accuracy. In Figure 3.5.1, the actual and estimated parameter values obtained from the simulation data are displayed. We also investigate the standard errors of the regression parameter estimates of the variance component model by comparing it with the robust estimate of standard deviation RESD($\hat{\beta}$). We give in Table 3.5.1 the mean and median of the estimated β and its estimated standard errors together with the RESD($\hat{\beta}$). Row value of β is considered the actual values from which $\zeta_{ij\ell}$ was simulated. Since, for all λ's values, the estimators have the same performance, only the results of one value of λ are presented in the table. If we look at Figure 3.5.1, the lines inside each box represent the median and the dots above and below the box are the outliers. A reference line is added to Figure 3.5.1 which indicates the actual value of 3 in which to display the position of the estimated parameter. For each boxplot, the parameter estimate is close to the true parameter value with some variation around the true value. Table 3.5.1 shows that

Median($\widehat{SE}(\hat{\beta})$) is nearly equal to RESD($\hat{\beta}$).

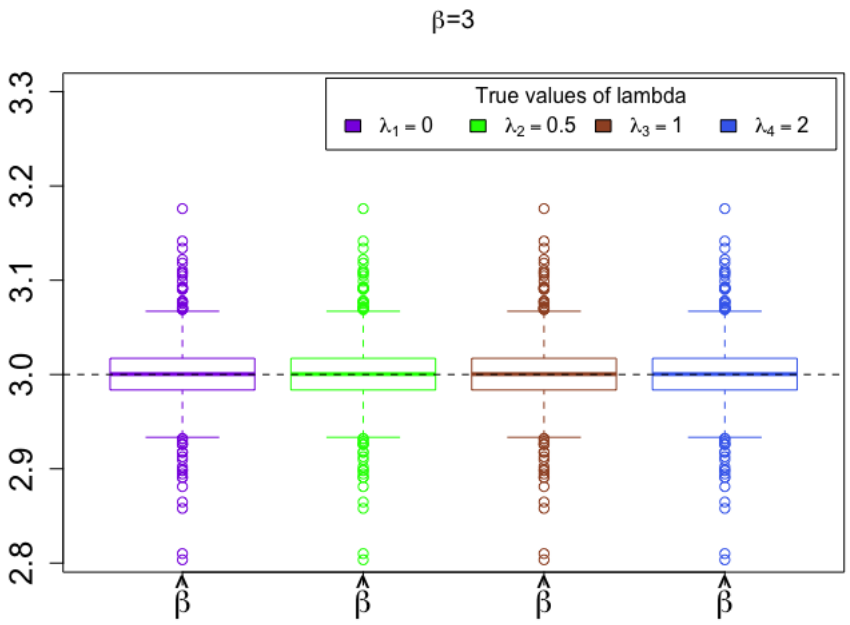


Figure 3.5.1: Simulation results: estimated β for fixed $\lambda_\ell = 0, 0.5, 1, 2$ with $K = 4$(from left to right).

β	3
Mean($\hat{\beta}$)	3.001097
Median($\hat{\beta}$)	3.000435
RESD($\hat{\beta}$)	0.0248572
Mean($\widehat{SE}(\hat{\beta})$)	0.02751033
Median($\widehat{SE}(\hat{\beta})$)	0.02153946

Table 3.5.1: Summary of simulation results for $\lambda = 0$

In the second case, it is more complex scenario, as we are estimating λ and β simultaneously using $K = 4$, yielding for each (true) value of λ a total of 1000 estimates of $\hat{\lambda}$ and $\hat{\beta}$.

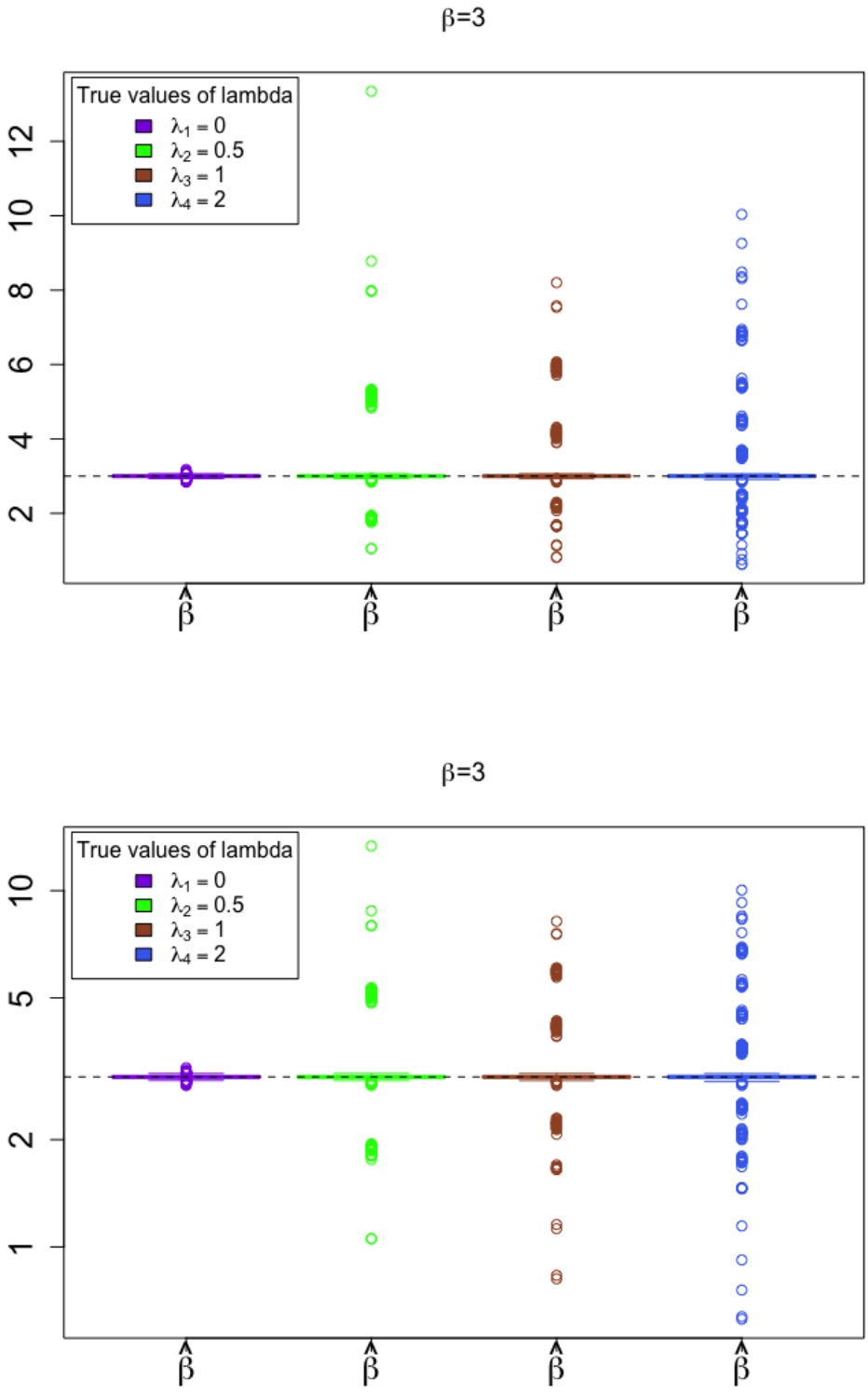


Figure 3.5.2: Simulation results: Estimates $\hat{\beta}$, in each plot for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right). The lower plot is exactly the upper plot with logarithmic scale in the vertical axis. Horizontal lines indicate the true values.

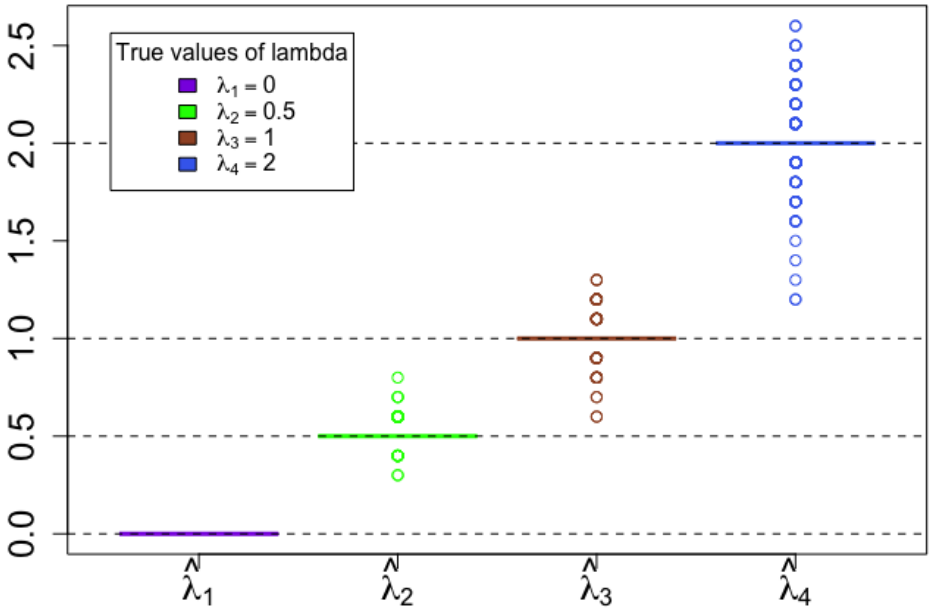


Figure 3.5.3: Simulation results: estimated λ, for true $\lambda_\ell = 0, 0.5, 1, 2$ (from left to right).

In Figures 3.5.2 and 3.5.3 we graph the boxplots for the regression and transformation parameters estimates, respectively. Again, the reference lines in the Figures indicate the actual values of the parameters. It is clear that the medians of the estimated β and λ is approximately equal to the true value in each plot. There are some outliers in each of the plots; in fact the outliers in the transformation estimates cause the outliers in the regression estimates as they shift the scale of the linear predictor. The medians of the estimated β and λ parameters are also provided in Table 3.5.2; we see that the medians for the transformation parameters sit exactly at their true values, and those of the regression parameters approximately so.

	$\lambda = 0$	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
Mean($\hat{\lambda}$)	0	0.5026	1.0028	2.0049
Median($\hat{\lambda}$)	0	0.5	1	2
β	3	3	3	3
Mean($\hat{\beta}$)	2.999629	3.0901	3.077008	3.109035
Median($\hat{\beta}$)	3.000329	3.000122	3.000329	3.000563
RESD($\hat{\beta}$)	0.02456314	0.02513099	0.02552865	0.0335325
Mean($\hat{SE}(\hat{\beta})$)	0.02563301	0.02674056	0.02643427	0.02683076
Median($\hat{SE}(\hat{\beta})$)	0.02140653	0.02137434	0.02137203	0.02141723

Table 3.5.2: Summary of simulation results using unknown values of λ

We see from Table 3.5.2 that the EM–based standard errors are close to their robust counterparts. The simulation results show that our approach is able to obtain estimates close to the real values of the parameters. Comparing this simulation to that for random effect models, there was much less variability in the estimates of the transformation and regression parameters of the variance components model. This is the reason for the standard errors to be better behaved here. Note that outliers were ignored by RESD($\hat{\beta}$), but $\hat{SE}(\hat{\beta})$ ignored the uncertainty due to the EM algorithm itself.

3.6 Applications

Example 3.6.1. the heights of boys in Oxford data

We are going to take another look at the `Oxboys` data that we have investigated earlier without transformations, in Example 3.2.1. The data has been explained in Figure

3.2.1. A further analysis of the model given in (3.2.14) is presented in this example using the Box–Cox transformation. We begin with the function `Kfind.boxcox()` which returns the optimal `tol`'s together with the disparities, AIC and BIC values for each class K, $K \in [2, 10]$:

R Note:

```
library(boxcoxmix)
testK <- Kfind.boxcox(height ~  age, groups = Oxboys$boy,
data = Oxboys, find.k = c(2,10), model.selection = "bic")
#Minimal BIC: 1019.743 at K= 9
```

K	1	2	3	4	5	6	7	8	9	10
optim `tol`	–	1.5	1.2	0.2	0.8	1.1	0.5	0.5	0.5	0.3
$-2\ell_P(\lambda)$	1641.93	1466.7617	1320.8801	1212.6595	1132.8487	1048.2698	1017.2692	931.3750	916.0921	908.0036
AIC	1649.875	1476.7617	1334.8801	1230.6595	1154.8487	1074.2698	1047.2692	965.3750	954.0921	950.0036
BIC	1658.296	1494.038	1359.067	1261.757	1192.857	1119.189	1099.099	1024.116	1019.743	1022.565

Table 3.6.1: Comparison of results from the untransformed `Oxboys` data ($\lambda = 1$), using K from 1 to 10

Concerning the choice of K for the untransformed data, it is transparent from Table 3.6.1 that there is a consistent improvement when increasing the number of mass points from $K = 1$ to $K = 9$. Aitkin et al. (2009) recommend the use of $K = 8$ mass points for this data set. Table 3.6.1 also shows the optimal `tol` that minimizes the disparity value for each K which can be then used in performing a grid search over λ to achieve the best results.

K	1	2	3	4	5	6	7	8	9	10
$\hat{\lambda}$	0.8625	-0.9375	0.1875	0.4125	-0.1500	-0.3325	0.4125	-0.1900	-0.5200	0.0750
$-2\ell_P(\hat{\lambda})$	1641.8793	1457.8640	1318.4705	1211.3445	1121.0861	1025.2499	1002.9139	887.4878	878.5608	866.7590
AIC	1649.8793	1467.8640	1332.4705	1229.3445	1143.0861	1051.2499	1032.9139	921.4878	916.5608	908.7590
BIC	1663.7005	1485.1406	1356.6578	1260.4424	1181.0946	1096.1691	1084.7437	980.2283	982.2119	981.3207

Table 3.6.2: Comparison of results from the transformed `Oxboys` data using K from 1 to 10

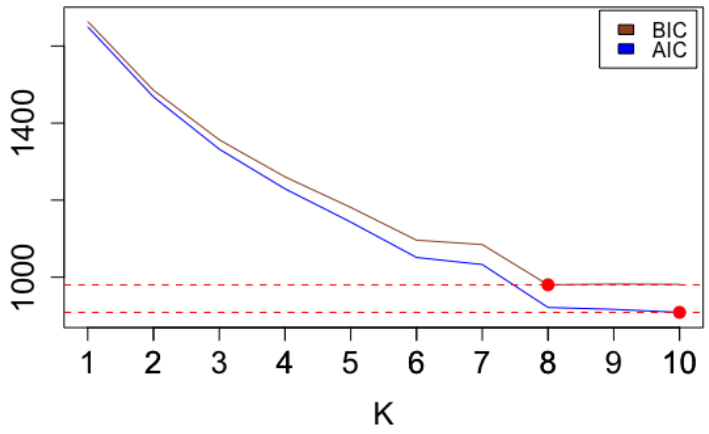


Figure 3.6.1: AIC and BIC values of the model after applying the response transformation to the `Oxboys` data for $K \in [1, 10]$

As measures of model fit, AIC and BIC values of the model after applying the response transformation for each class are presented in Table 3.6.2 and Figure 3.6.1. BIC criteria indicates that an 8–component model is the best choice, even though 10 component are necessary to reduce the disparity ($-2\ell_P(\hat{\lambda})$) and hence maximize the non-parametric profile log–likelihoods ($\ell_P(\lambda)$) equation given in (3.3.22). The results before and after applying the response transformation are summarized in Table 3.6.1 and Table 3.6.2. As can be seen from these tables, comparing AIC and BIC values of the untransformed model fit ($\lambda = 1$) and our method using $K = 1, ..., 10$, respectively, showed a better performance of the NPPML approach. In other words, using the response after applying the transformation leads to a better

fitting model than the original data. For the specified range of K from 1 to 10, $\hat{\lambda}$'s values that maximize $\ell_P(\lambda)$'s are shown in Figure 3.6.2. We can see here a different behaviour between K and λ. Until we include more than one component there is no strong evidence that transformation works, but once we go to 3 or more classes we see that a log-transformation ($\lambda = 0$) is not so far off the mark and to some extent intuitively appealing with `height` $\propto$ `age`. The minimal BIC value occurred at K=8 with $\hat{\lambda} = -0.19$ and *tol*= 0.5 (BIC=980.2283), indicating a better fit. Figure 3.6.2 also shows a strong need of a transformation as we increase the number of classes. This provides additional evidence for a better fit to the transformed data.

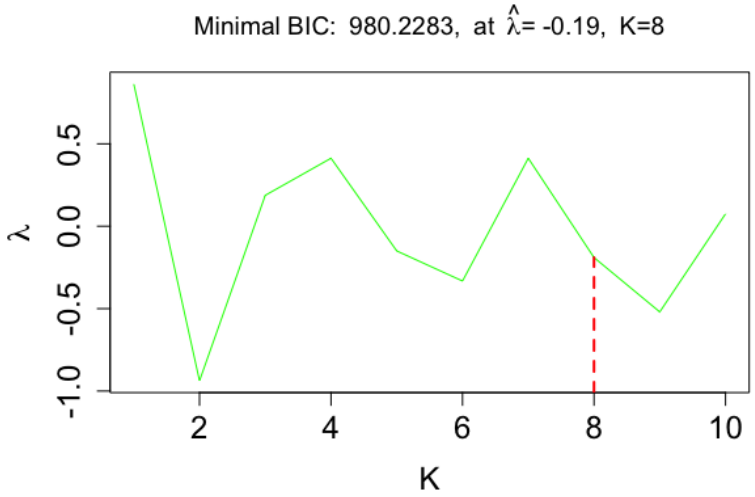


Figure 3.6.2: $\hat{\lambda}$ as a function of K with the optimal `tol` for each class of the `Oxboys` data

R Note:

```
fitk8 <- np.boxcoxmix(height ~  age, groups = Oxboys$boy,
data = Oxboys, K = 8, tol =0.5, start = "gq", lambda=-0.19)
```

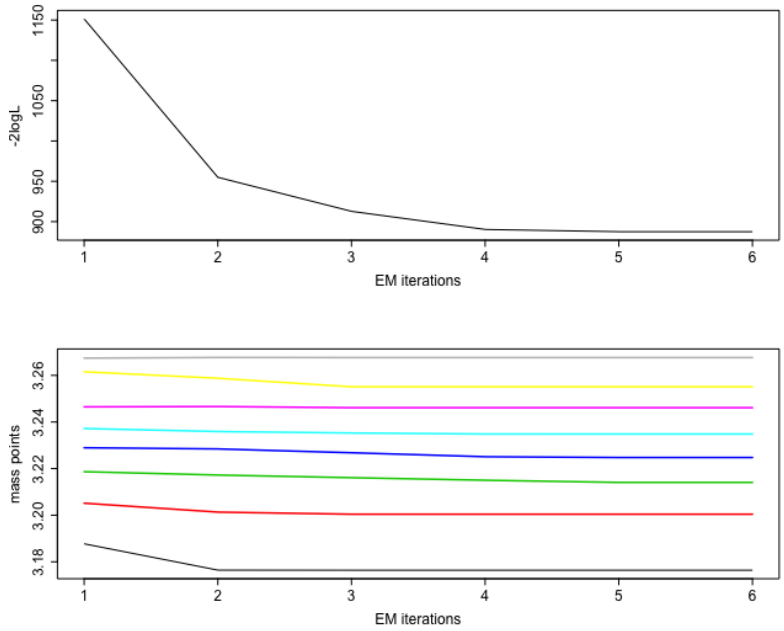


Figure 3.6.3: The disparities $(-2\log L)$ against EM iteration number for the 8 mass–points transformed model of the `Oxboys` data using $\lambda = -0.19$ and $tol = 0.5$.

The posterior probabilities for the transformed data with 8 mixture classes are plotted in Figure 3.6.3, the distance between the classes is clearly visible, indicating that the posterior probabilities are all nicely converged to either 0 or 1 (or close to it).

3.7 Discussion

In this Chapter, we have applied the Box–Cox transformation to the variance component models using NPPML approach in the same fashion as in the random effects model. Two simulation scenarios are carried out to assess the performance of our approach using fixed and unknown values of the transformation parameters by the same simulation process and estimation method which were used for random effects

model. To some extent, the results of this simulation differ from those of the random effects models. As expected, these results have lower variability in the transformation and regression parameters estimates, yielding more accurate standard errors because there is much more information about the random effects as these are now shared between lower–level units in each upper-level units, rather than being unique to each observation as in the random effect models considered in Chapter 2 (see Aitkin et al. (2005)).

As we noted in Chapter 2, the variability of $\hat{\lambda}$ increases for larger values of λ and this variability is the main cause of the variation in the regression parameters estimates. Similar to the case in the random effects model, there is a strong consistency of the parameter estimates when $\lambda = 0$ is the most suitable transformation for the simulated data. For this simulation, we have used 20 clusters with a length of 5 units for each cluster. For future research, the effect of a large number of clusters in the estimation results of the proposed approach should be examined. Another line of future investigation would be studying the impact of adding further explanatory variables to our simulation model.

Additionally, searching for the minimizer of the model selection criteria AIC and BIC for the `Oxboys` data found that all transformed models using $\hat{\lambda}$ obtained from our proposed approach yielded substantially better fits than the models without transformation at all. Also for this data, there is a strong need of a transformation as we increase the number of components as shown in Figure 3.6.2 that provides a further support for the needs of considering the two–level random effects model in such transformation.

Chapter 4

Transformations for logistic regression models

4.1 Introduction

The logistic regression model is a statistical technique that is well suited to applied statistical analyses. It is commonly used to model and solve classification problems. To analyze the dependence of binary response on predictor variables it is common to connect the success probabilities to the linear predictors through a link function. In such case, a common link function is the logistic (logit) function that transforms the interval $(0, 1)$ to $(-\infty, \infty)$. We will employ the Box–Cox transformations that include the logistic and the power transformations to carry out the analysis of binary response. This approach allows the data to meet its needs via a suitable transformation that gives a simpler or better fit. Aranda-Ordaz (1981) applied parametric family of transformations to the success probability in order to achieve

additivity for binary response data. Guerrero and Johnson (1982) suggested to apply the Box–Cox transformation to the odds–ratio to generalize the logistic model. In this Chapter, our proposal is similar to the Guerrero and Johnson (1982) idea but is implemented in a different way.

Section 4.2 provides an introduction to the binary logistic regression models. In Section 4.3, the Box-Cox transformation is applied to the binary regression model, along with a software description in Section 4.4. The applicability of the proposed approach is demonstrated by simulated and real data examples in Sections 4.5 and 4.6. Finally, we provide a discussion in Section 4.7.

4.2 Logistic regression model

Binary logistic regression is a method that is used to assess the associations between a set of independent variables and a single binary dependent variable. When the response variable is binary, then the probability distribution of the number of successes in a sample of a particular size, for given values of the predictor variables, is called a Bernoulli distribution. Here, we assume the response Y_i, $i = 1, \ldots, n$, follows a binomial distribution with $Y_i \sim B(m_i, P_i)$ where m_i is the number of trials and P_i is a vector with fixed success probabilities for each trial within each category. The probability distribution function of Y_i is given by

$$f(Y_i = y_i) = \binom{m_i}{y_i} P_i^{y_i}(1 - P_i)^{m_i - y_i} \tag{4.2.1}$$

where $y_i = 0, 1, ..., m_i$, $P_i^{y_i}(1-P_i)^{m_i-y_i}$ is the probability of having $m_i - y_i$ failures and y_i successes in a particular order, and the number of ways of observing y_i successes

in m_i trials is given by the binomial coefficient. It follows that $E(Y_i) = m_i P_i$ and $var(Y_i) = m_i P_i(1 - P_i)$. If $m_i = 1$, then Y_i follows a Bernoulli distribution with mean and variance as $E(Y_i) = P_i$ and $var(Y_i) = P_i(1 - P_i)$, respectively. Note that, for both cases the mean and the variance depend on the probability of the responses P_i. Factors affecting this probability will alter both the mean and the variance of the observations (Rodriguez, 2012).

4.2.1 The logit link function

To get the logit of the probability, we make a transformation of the linear probability function $P_i = \eta_i$, where $\eta_i = x_i^T \beta$ is the linear predictor (Rodriguez, 2012). The difficulty is that the linear predictor η_i on the right-hand-side can be any real number, but the probability P_i on the left-hand-side is bounded between the values of zero and one. To solve this, first, we express the probability of the response in terms of *odds* as

$$\text{odds}_i = \frac{P_i}{1 - P_i} = e^{\eta_i}, \tag{4.2.2}$$

which maps P_i from $[0, 1]$ to $[0, \infty]$. Second, if we take the logarithms of the *odds*, then $\log(P_i/1 - P_i)$ goes from $-\infty$ to ∞. The mathematical form of the logit transformation follows from (4.2.2),

$$\text{logit}(P_i) = \log\left(\frac{P_i}{1 - P_i}\right) = \eta_i. \tag{4.2.3}$$

Note that P_i and $1 - P_i$ have to be positive and so do the odds. As $P_i \to 0$, $P_i/(1 - P_i) \to 0$ and as $P_i \to 1$, $P_i/(1 - P_i) \to \infty$. Now, the logistic function in

terms of the probability of the response is called the inverse of the transformation:

$$P_i = logit^{-1}(\eta_i) = \frac{e^{\eta_i}}{1+e^{\eta_i}} = \frac{1}{1+e^{-\eta_i}} \tag{4.2.4}$$

The cumulative distribution function (cdf) for the binomial distribution is $F(\eta_i) = P_i$, for $-\infty < \eta_i < \infty$.

4.2.2 Maximum likelihood estimation of the regression parameters

The likelihood function for n independent observations is a product of probability densities given in (4.2.1). The log-likelihood can be written as

$$\ell = \log L = \sum_{i=1}^{n} \log f(y_i) \tag{4.2.5}$$

where

$$\begin{aligned} \log f(y_i) &= y_i \log P_i + (m_i - y_i)\log(1-P_i) \\ &= m_i \log(1-P_i) + y_i \log \frac{P_i}{(1-P_i)} \\ &= -m_i \log(1+e^{\eta_i}) + y_i\eta_i \end{aligned} \tag{4.2.6}$$

then

$$\ell = \sum_{i=1}^{n} \Big[-m_i \log(1+e^{\eta_i}) + y_i\eta_i \Big] \tag{4.2.7}$$

The maximum likelihood estimator of the parameters are those values of $\hat{\beta}$ that maximize the likelihood function. The estimate of β can be obtained by setting

the Score vector

$$\frac{\partial \ell}{\partial \beta_j} = 0 \qquad , j = 0, 1, ..., r,$$

where

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{n} x_{ij} \Big[- m_i P_i + y_i \Big], \tag{4.2.8}$$

and solve for β. The standard errors of the parameter estimates, $\hat{\beta}$, are computed using the information matrix which is obtained by taking the second derivatives of the log-likelihood function with respect to β.

4.2.3 Existing R implementation: `glm()`, `alldist()`

To compute the parameter estimates of the logistic (logit) model, we can use the available statistical packages in R. The **npmlreg** (Einbeck et al., 2014) function `alldist()` can fit the aforementioned model by setting $k = 1$ and `family = binomial(link="logit")`. The results of which are identical to those of the **stats** (R Core Team, 2016) function `glm()`. For real data examples see (Dalgaard, 2008, p. 226) and (Rao and Rao, 2014, p. 257).

4.3 Transformations for binary regression models

In this section, the Box-Cox transformation is extended to the logistic regression model. Guerrero and Johnson (1982) assumed that the power transformation of the

odds ratio results in a linear model. That is

$$\Big\{P_i/(1-P_i)\Big\}^{(\lambda)} = x_i^T\beta = \eta_i \quad , i = 1, ..., n. \tag{4.3.1}$$

The Box-Cox transformation of the odds–ratio is thus,

$$\Big\{P_i/(1-P_i)\Big\}^{(\lambda)} = \begin{cases} \dfrac{\Big\{P_i/(1-P_i)\Big\}^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log\Big\{P_i/(1-P_i)\Big\} & (\lambda = 0) \end{cases} \tag{4.3.2}$$

where the restrictions $0 < P_i < 1$ and $P_i/(1-P_i) > 0$ apply. Note that the Box-Cox transformation here is used as a parametric link function that is very different from the response transformation in the previous chapters. Now for $\lambda \neq 0$

$$\begin{aligned}
\frac{\Big\{P_i/(1-P_i)\Big\}^{\lambda} - 1}{\lambda} &= \eta_i \\
\Big\{P_i/(1-P_i)\Big\}^{\lambda} - 1 &= \lambda\eta_i \\
\Big\{P_i/(1-P_i)\Big\}^{\lambda} &= 1 + \lambda\eta_i \\
P_i/(1-P_i) &= \Big(1+\lambda\eta_i\Big)^{1/\lambda} \\
P_i &= (1-P_i)\Big(1+\lambda\eta_i\Big)^{1/\lambda} \\
P_i &= \Big(1+\lambda\eta_i\Big)^{1/\lambda} - P_i\Big(1+\lambda\eta_i\Big)^{1/\lambda} \\
P_i + P_i\Big(1+\lambda\eta_i\Big)^{1/\lambda} &= \Big(1+\lambda\eta_i\Big)^{1/\lambda} \\
P_i\Big(1+\Big(1+\lambda\eta_i\Big)^{1/\lambda}\Big) &= \Big(1+\lambda\eta_i\Big)^{1/\lambda} \\
P_i &= \frac{\Big(1+\lambda\eta_i\Big)^{1/\lambda}}{1+\Big(1+\lambda\eta_i\Big)^{1/\lambda}}
\end{aligned}$$

$$= \frac{1}{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1}$$

$$\implies P_i = \left\{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1\right\}^{-1} \tag{4.3.3}$$

and

$$\begin{aligned}
1-P_i &= 1-\left\{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1\right\}^{-1} \\
&= 1-\frac{1}{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1} \\
&= \frac{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1-1}{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1} \\
&= \frac{\left(1+\lambda\eta_i\right)^{-1/\lambda}}{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1} \\
&= \frac{1}{1+\left(1+\lambda\eta_i\right)^{1/\lambda}} \\
\implies 1-P_i &= \left\{\left(1+\lambda\eta_i\right)^{1/\lambda}+1\right\}^{-1}
\end{aligned} \tag{4.3.4}$$

Now

$$\begin{aligned}
\frac{P_i}{1-P_i} &= \frac{\left\{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1\right\}^{-1}}{\left\{\left(1+\lambda\eta_i\right)^{1/\lambda}+1\right\}^{-1}} \\
&= \frac{\left\{\left(1+\lambda\eta_i\right)^{1/\lambda}+1\right\}}{\left\{\left(1+\lambda\eta_i\right)^{-1/\lambda}+1\right\}} \\
&= \left(1+\lambda\eta_i\right)^{1/\lambda}\frac{\left\{\left(1+\lambda\eta_i\right)^{1/\lambda}+1\right\}}{\left\{\left(1+\lambda\eta_i\right)^{1/\lambda}+1\right\}} \\
\implies \frac{P_i}{1-P_i} &= \left(1+\lambda\eta_i\right)^{1/\lambda}
\end{aligned} \tag{4.3.5}$$

It follows from Equation (4.3.5) and the general properties of the exponential function that

$$\lim_{\lambda \to 0} \Big(1 + \lambda \eta_i\Big)^{1/\lambda} = e^{\eta_i}.$$

Also, from Equation (4.3.3), P_i approaches 1 as λ approaches $\pm\infty$. Note that λ and η_i should be both positive values or both negative values to avoid having roots of negative numbers in the odds, and hence the probability. From the inversion of (4.3.2) we get

$$\tilde{P}_i = \begin{cases} \Big\{\Big(1 + \lambda \eta_i\Big)^{-1/\lambda} + 1\Big\}^{-1} & (\lambda \neq 0), \\ \Big\{1 + e^{-\eta_i}\Big\}^{-1} & (\lambda = 0) \end{cases} \tag{4.3.6}$$

4.3.1 Maximum likelihood estimation of the regression parameters

The probability density function of Y_i is given by

$$f(y_i) = \binom{m_i}{y_i} \tilde{P}_i^{y_i} (1 - \tilde{P}_i)^{m_i - y_i} \tag{4.3.7}$$

The likelihood function can be written as

$$L(\lambda, \beta) = \prod_{i=1}^{n} f(y_i) \tag{4.3.8}$$

The log-likelihood is thus

$$\ell = \log L = \sum_{i=1}^{n} \log f(y_i) \tag{4.3.9}$$

where

$$\begin{aligned}
\log f(y_i) &= y_i \log \tilde{P}_i + (m_i - y_i) \log\Big(1 - \tilde{P}_i\Big) \\
&= m_i \log\Big(1 - \tilde{P}_i\Big) + y_i \log\Big\{\tilde{P}_i/(1 - \tilde{P}_i)\Big\} \\
&= -m_i \log\Big(\Big(1 + \lambda\eta_i\Big)^{1/\lambda} + 1\Big) + y_i \log\Big(1 + \lambda\eta_i\Big)^{1/\lambda} \\
&= -m_i \log\Big(\Big(1 + \lambda\eta_i\Big)^{1/\lambda} + 1\Big) + \frac{y_i}{\lambda} \log\Big(1 + \lambda\eta_i\Big) \qquad (4.3.10)
\end{aligned}$$

then

$$\ell = \sum_{i=1}^{n} \Big[- m_i \log\Big(\Big(1 + \lambda\eta_i\Big)^{1/\lambda} + 1\Big) + \frac{y_i}{\lambda} \log\Big(1 + \lambda\eta_i\Big)\Big] \qquad (4.3.11)$$

Note that, in contrast to the linear model in Chapters 2 and 3, no Jacobian is needed for this case because the transformation does not act on the distribution of the data. The estimate of β can be obtained by taking the derivative of Equation (4.3.11) with respect to β, setting it equal to zero and solve for β.

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{n} \left[-m_i \frac{\lambda x_i (1/\lambda)\Big(1 + \lambda\eta_i\Big)^{(1/\lambda)-1}}{\Big(1 + \lambda\eta_i\Big)^{1/\lambda} + 1} + \frac{y_i}{\lambda} \frac{\lambda x_i}{\Big(1 + \lambda\eta_i\Big)} \right] \\
&= \sum_{i=1}^{n} \left[-m_i \frac{x_i \Big(1 + \lambda\eta_i\Big)^{1-\lambda/\lambda}}{\Big(1 + \lambda\eta_i\Big)^{1/\lambda} + 1} + y_i x_i \Big(1 + \lambda\eta_i\Big)^{-1} \right] \\
&= \sum_{i=1}^{n} \left[-m_i \frac{x_i \Big(1 + \lambda\eta_i\Big)^{-1}}{\Big(1 + \lambda\eta_i\Big)^{-1/\lambda} + 1} + y_i x_i \Big(1 + \lambda\eta_i\Big)^{-1} \right] \\
&= \sum_{i=1}^{n} x_i \Big(1 + \lambda\eta_i\Big)^{-1} \Big[-m_i \Big\{\Big(1 + \lambda\eta_i\Big)^{-1/\lambda} + 1\Big\}^{-1} + y_i \Big] \\
\Longrightarrow \frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{n} x_i \Big(1 + \lambda\eta_i\Big)^{-1} \Big[- m_i \tilde{P}_i + y_i \Big] \qquad (4.3.12)
\end{aligned}$$

Replacing the results into Equation (4.3.11) we get the profile log-likelihood function.

$$\ell_P(\lambda) = \sum_{i=1}^{n} \Big[- m_i \log\Big(\big(1+\lambda\hat{\eta}_i\big)^{1/\lambda} + 1\Big) + \frac{y_i}{\lambda}\log\big(1+\lambda\hat{\eta}_i\big)\Big] \tag{4.3.13}$$

The profile maximum likelihood estimate of λ is therefore given by

$$\hat{\lambda} = arg \max_{\lambda} \ \ell_P(\lambda). \tag{4.3.14}$$

which can be found through a grid search over λ. In practice, we maximize Equation (4.3.13) over a given grid of values for λ using an iterative process between the estimation of $\beta^{(\lambda)}$.

4.3.2 Estimation of the transformation parameter

Here, we are seeking for a direct way of obtaining NPPML estimate of the transformation parameter by deriving equation (4.3.11) with respect to λ as follows

$$\begin{aligned}
\frac{\partial \ell}{\partial \lambda} &= \sum_{i=1}^{n} \Bigg[- m_i \frac{\big(1+\lambda\eta_i\big)^{1/\lambda}\Big(\dfrac{\eta_i}{\lambda\big(1+\lambda\eta_i\big)} - \dfrac{\log\big(1+\lambda\eta_i\big)}{\lambda^2}\Big)}{\big(1+\lambda\eta_i\big)^{1/\lambda}+1} \\
&\qquad + y_i\Big(\frac{\eta_i}{\lambda\big(1+\lambda\eta_i\big)} - \frac{\log\big(1+\lambda\eta_i\big)}{\lambda^2}\Big)\Bigg] \\
&= \sum_{i=1}^{n} \Big(\frac{\eta_i}{\lambda\big(1+\lambda\eta_i\big)} - \frac{\log\big(1+\lambda\eta_i\big)}{\lambda^2}\Big)\Bigg[- m_i \frac{\big(1+\lambda\eta_i\big)^{1/\lambda}}{\big(1+\lambda\eta_i\big)^{1/\lambda}+1} + y_i\Bigg] \\
&= \sum_{i=1}^{n} \Big(\frac{\eta_i}{\lambda\big(1+\lambda\eta_i\big)} - \frac{\log\big(1+\lambda\eta_i\big)}{\lambda^2}\Big)\Bigg[- m_i \frac{1}{\big(1+\lambda\eta_i\big)^{-1/\lambda}+1} + y_i\Bigg] \\
&= \sum_{i=1}^{n} \Big(\frac{\eta_i}{\lambda\big(1+\lambda\eta_i\big)} - \frac{\log\big(1+\lambda\eta_i\big)}{\lambda^2}\Big)\Bigg[- m_i \Big\{\big(1+\lambda\eta_i\big)^{-1/\lambda}+1\Big\}^{-1} + y_i\Bigg]
\end{aligned}$$

$$= \frac{1}{\lambda}\sum_{i=1}^{n}\left(\eta_i\big(1+\lambda\eta_i\big)^{-1} + \lambda^{-1}\log\big(1+\lambda\eta_i\big)^{-1}\right)\left[-m_i\tilde{P}_i + y_i\right] \tag{4.3.15}$$

There is no analytical solution for the estimate of λ given that $\dfrac{\partial \ell}{\partial \lambda} = 0$. Therefore, we do not consider it further in this thesis. However, it can be solved numerically. Guerrero and Johnson (1982) suggested obtaining the ML estimates by setting Equations (4.3.12) and (4.3.15) equal to zero and solving for β and λ by a quasi-Newton procedure.

4.4 Software description

The proposed approach is implemented for fixed λ as a link function which is applicable to both functions `glm()` and `alldist()`, by setting `family = binomial(link= boxcoxpower(Lambda))` where `Lambda` can be any value. However, one can perform a grid search over λ using the **boxcoxmix** (Almohaimeed and Einbeck, 2017) function `boxcoxtype()` with $k = 1$. We will experiment this method on simulated and real data sets in the two next sections. The R code for the **boxcoxmix** link function `boxcoxpower(Lambda)` which is an implementation of the Box–Cox transformation and its inverse given in Equations (4.3.2) and (4.3.6), respectively.

R Note:

```
boxcoxpower <- function(Lambda=0)
{
```

```
  linkfun <- function(mu){   if(Lambda==0) log(mu/(1-mu))
    else  (((mu/(1-mu))^Lambda)-1)/Lambda}
  linkinv <- function(eta) {  if(Lambda==0)  plogis(eta)
    else (((1+Lambda*eta)^(-1/Lambda))+1)^(-1)}
  mu.eta<- function(eta) { if(Lambda==0)
   ifelse(abs(eta)>30,.Machine$double.eps,
      exp(eta)/(1+exp(eta))^2) else
        pmax(((1+ Lambda*eta)^((1/Lambda) -1))/
        ((1+ Lambda*eta)^(1/Lambda) +1)^2, .Machine$double.eps)}

  valideta <- function(eta) TRUE
  link <-paste("boxcoxpower(",Lambda,")", sep="")
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta,
                 name = link),
            class = "link-glm")
}
```

4.5 Simulation study

In this Section, we conduct two simulation studies. First one is based on the comparison of logistic model and Box-Cox-type models, and the second study is based on

investigating the ability of the proposed methods in estimating the transformation and regression parameters simultaneously. The procedure used for the simulation studies is given in the appendix, Figure A.3.2.

Simulation Study 1

To assess the accuracy of our approach, we first simulate data by applying the Box–Cox transformation 'backwards' to a success probability P_i using a set of λ values. Specifically, for each of five given values λ_ℓ, $\ell = 1, 2, 3, 4, 5$, we generate 1000 datasets with 100 observations as follows,

$$y_{i\ell} \sim B(40, \tilde{P}_i(\lambda_\ell)), \qquad i = 1, ..., 100, \tag{4.5.1}$$

$$\tilde{P}_i(\lambda_\ell) = \begin{cases} \left\{\left(1 + \lambda_\ell \eta_i\right)^{-1/\lambda_\ell} + 1\right\}^{-1} & (\lambda_\ell \neq 0), \\ \left\{1 + e^{-\eta_i}\right\}^{-1} & (\lambda_\ell = 0) \end{cases}$$

$$\eta_i = 2 + x_i$$

$$X \sim U(-1, 1)$$

$$\lambda_1 = -0.2, \;\; \lambda_2 = 0, \;\; \lambda_3 = 0.2, \;\; \lambda_4 = 0.5, \;\; \lambda_5 = 1$$

Note that $\tilde{P}_i(\cdot)$ denotes the 'backward' Box–Cox–transformation. Refer to Appendix A for the R code that was used for generating these simulated data sets. In the estimation method, we apply the Box–Cox transformation forwards to the odds–ratio using fixed value of λ via the function `alldist()` with $k = 1$ and `family = binomial(link= boxcoxpower(Lambda))`, where `Lambda` is the same λ_ℓ that has been used in the simulation process for each dataset.

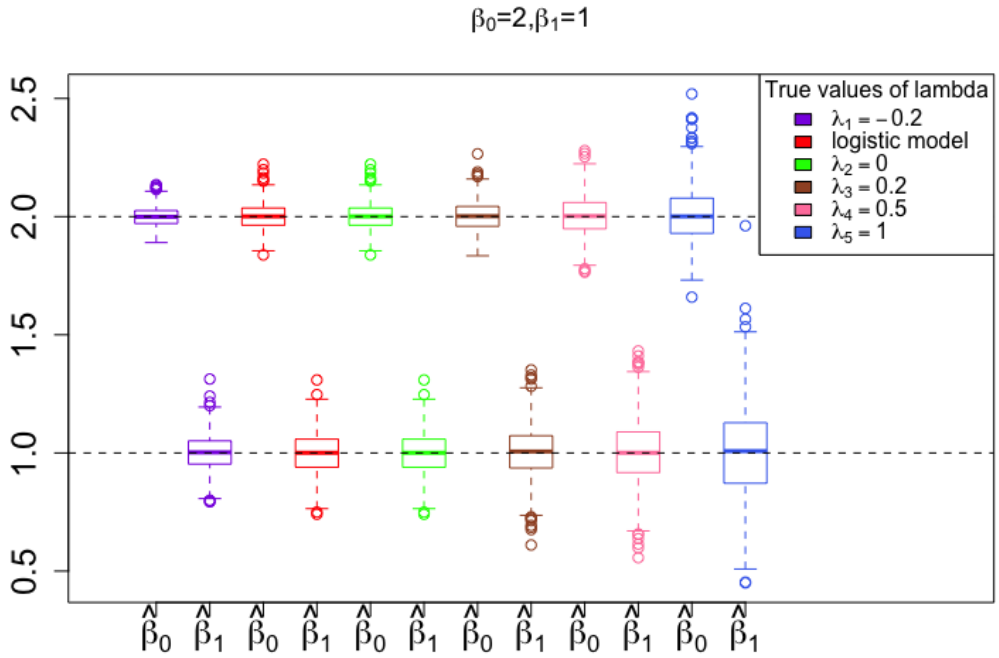


Figure 4.5.1: Simulation results: estimated β for fixed λ compared with logistic model, from left to right: $\lambda_1 = -0.2$, logistic model, $\lambda_\ell = 0, 0.2, 0.5, 1$, respectively.

Figure 4.5.1 shows the boxplots for logistic model coloured by red and the rest for the Box–Cox–type models with fixed λ as given in (4.5.1), to compare the results of the `logit` and `boxcoxpower(Lambda)` links functions. In this example, Figure 4.5.1 reflects the estimated values of β per 1000 simulations. We added reference lines in the boxplots which indicate the actual values of $\beta = (2, 1)$ to display the position of the estimated β for each boxplot. One can see that the the boxplots of the logistic and Box–Cox–type with $\lambda = 0$ are exactly the same. However, the rest of λ values reflect some interesting differences. Although the median of all of the boxplots captured the actual values of β's, the variation around the median of the boxplots can be compared. The boxplots of the Box–Cox–type with $\lambda = -0.2$ displays estimates that have less variation than any of the other plots. It is also

clear that there is increased variability in the β estimates as λ increases. This may suggest that the proposed link works better with negative λ values. However, as we discussed before, the odds (and hence probability) depend on λ and η_i. If λ or η_i has a negative value, then the resulting odds (and hence probability) can be undefined as it would be a root of a negative value.

Simulation Study 2

In this case, we are interested in examining the ability of our approach to estimate the transformation and regression parameters simultaneously through the function `boxcoxtype()` by performing a simulation and analyzing the results. The structure of the data in this study is the same as in (4.5.1) and the simulation process is conducted along the same lines as in the previous study, except that we now use unknown values of λ in the estimation method rather than fixed. We estimate λ by applying a grid search over λ and estimate β using the obtimal value of λ, yielding for each (true) value of λ a total of 1000 estimates of $\hat{\lambda}$ and $\hat{\beta}$. Figure 4.5.2 shows the estimates $\hat{\beta}$ for true $\lambda_\ell = -0.2, 0, 0.2, 0.5, 1$, respectively. The horizontal lines in the boxplots represent the actual values of β = (2,1). Obviously the medians of the estimated β's are approximately matching their true values with more variability as λ becomes larger. However, when $\lambda = 0$, the boxplots of $\hat{\beta}$ have more outliers than any of the other plots. A plot of the estimated λ appears in Figure 4.5.3 with horizontal lines indicating the actual values of λ = (-0.2,0,0.2,0.5,1). In each boxplot, it is clear that the median of the estimated λ tends to be very close to the true underlying value with an increase in variance as λ becomes larger.

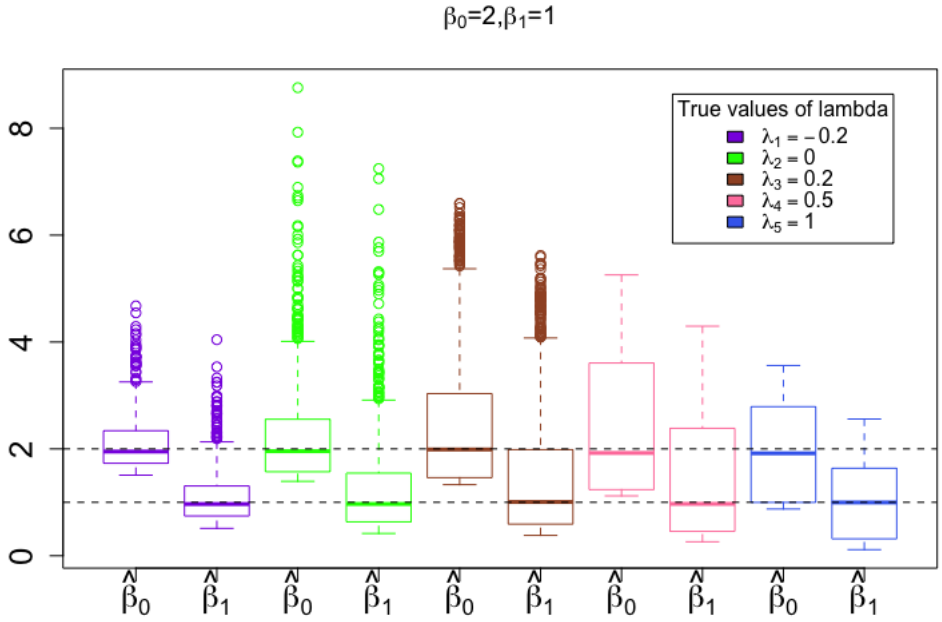


Figure 4.5.2: Simulation results: Estimates $\hat{\beta}$, in each plot for true $\lambda_\ell = -0.2, 0, 0.2, 0.5, 1$ (from left to right).

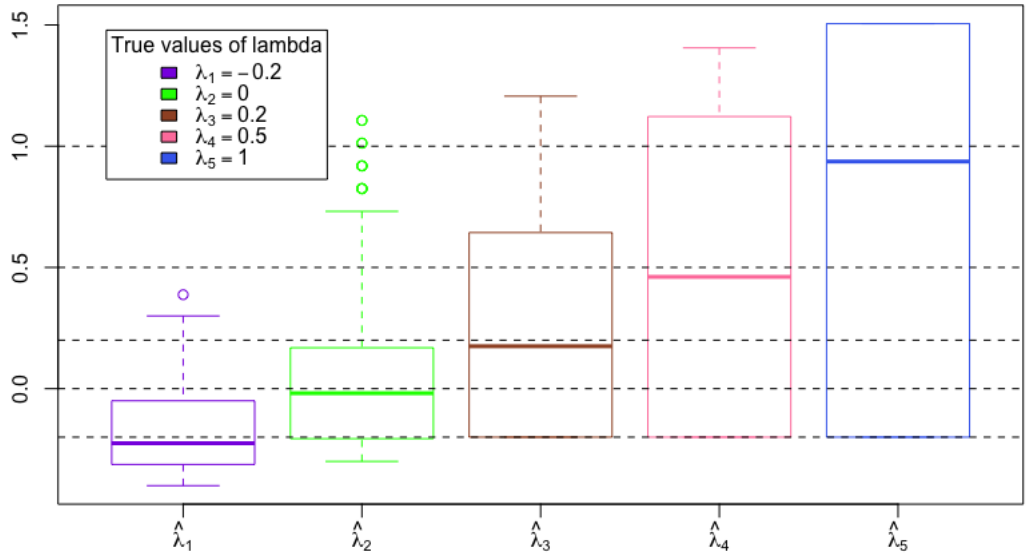
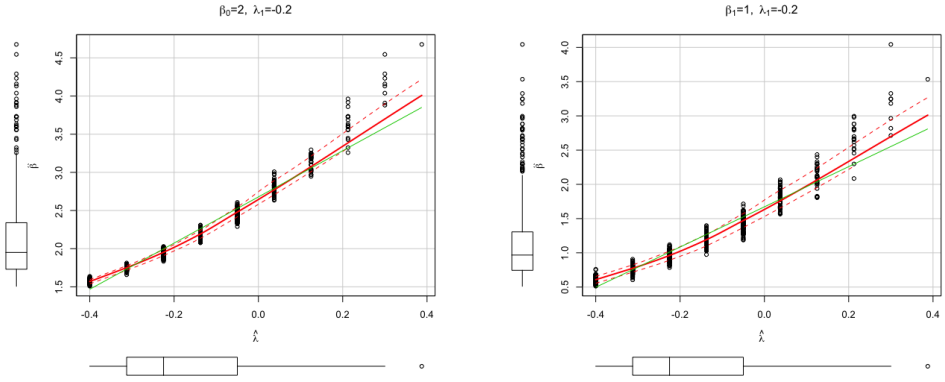
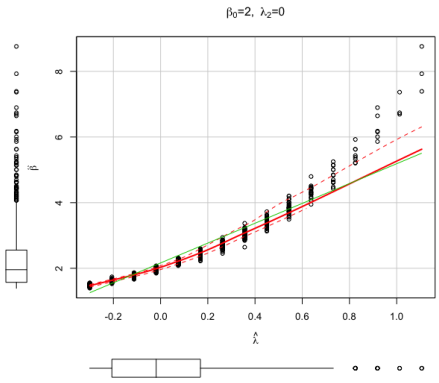


Figure 4.5.3: Simulation results: estimated λ, for true $\lambda_\ell = -0.2, 0, 0.2, 0.5, 1$ (from left to right).

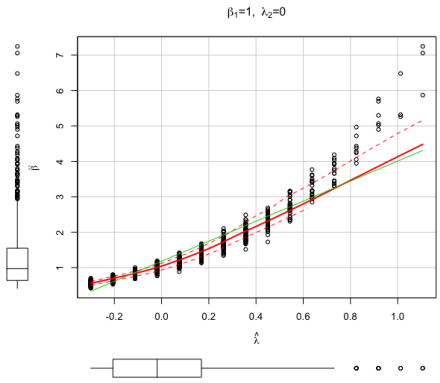


(a)

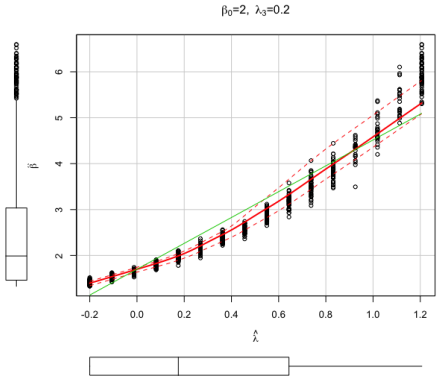
(b)



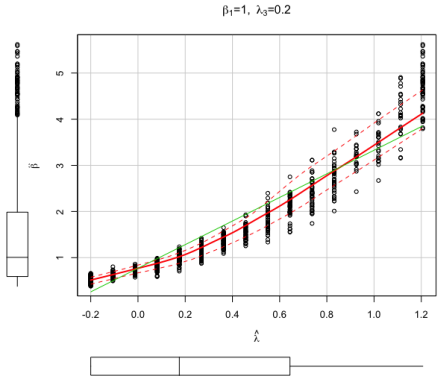
(c)



(d)

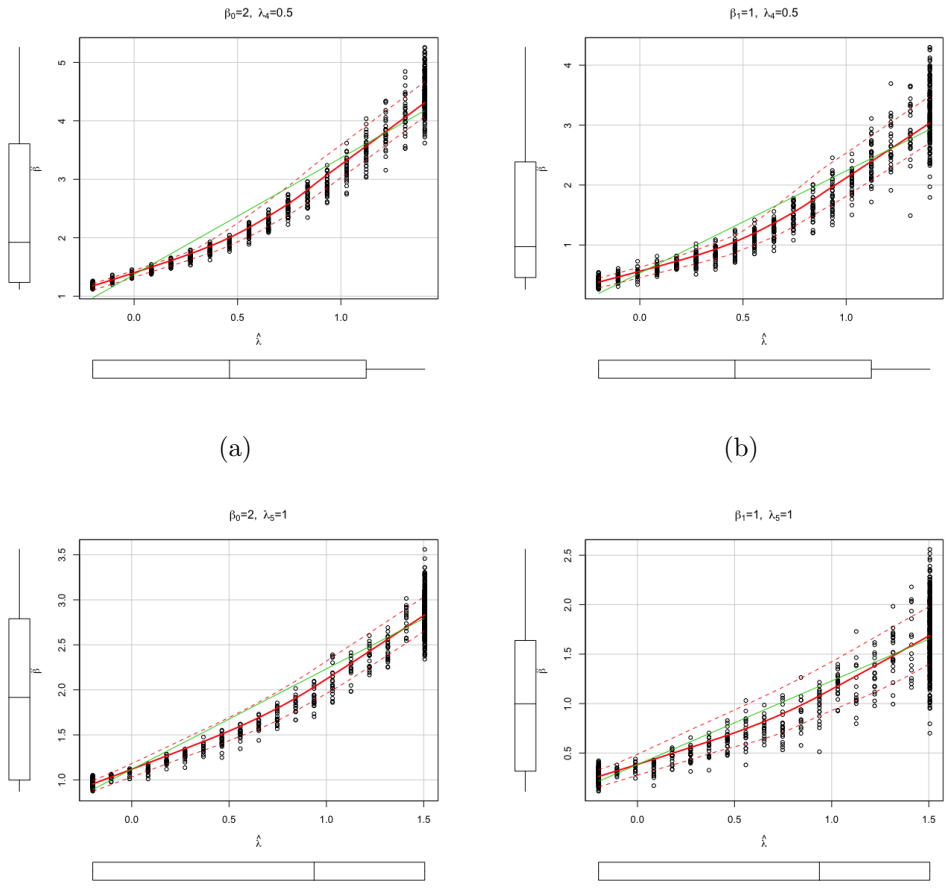


(e)



(f)

Figure 4.5.4: Simulation results: estimated regression parameters against estimated transformation parameters. In each plot for true $\beta = 2, 1$ (from left to right) and for true $\lambda = -0.2, 0, 0.2$ (from top to bottom),



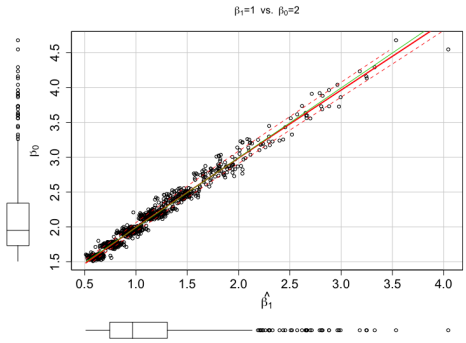
(a) (b)

(c) (d)

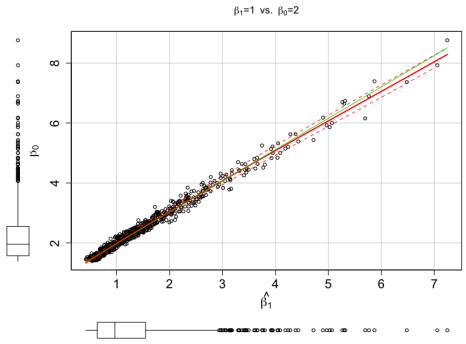
Figure 4.5.5: Simulation results: estimated regression parameters against estimated transformation parameters. In each plot for true $\beta = 2, 1$ (from left to right) and from top to bottom, $\lambda = 0.5, 1$.

Figures 4.5.4 and 4.5.5 show scatterplots of the estimated regression parameters $\hat{\beta}$ against the estimated transformation parameters $\hat{\lambda}$, in each plot for true $\beta_j = 2, 1,\ j = 0, 1$ (from left to right) and $\lambda_\ell = -0.2, 0, 0.2, 0.5, 1$ (from top to bottom). The boxplots in the margins of each scatterplot visualize the position and spread of the estimated parameters whereas the regression line inside the scatterplot explains the trend of the points to help us better understand the statistical relationship between the two parameters. We observe that the median of each boxplot is very close to the true underlying values of the two parameters β and λ. We also

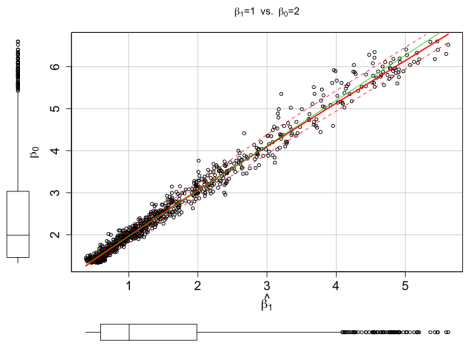
find that, for all scatterplots, there are nearly linear relationships between $\hat{\lambda}$ and $\hat{\beta}$ and the variances about the regression lines seem relatively constant, meaning that the transformation parameters λ influence the regression parameters β. As in the previous Chapters, the outliers in the transformation estimates cause the outliers in the regression estimates as they shift the scale of the linear predictor.



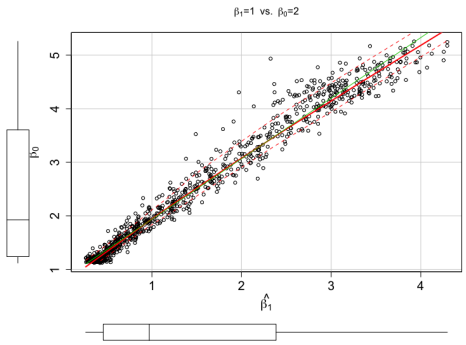
(a) $\lambda = -0.2$



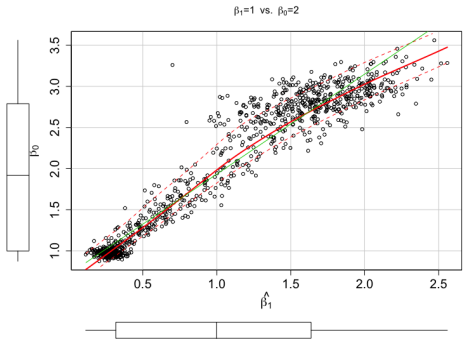
(b) $\lambda = 0$



(c) $\lambda = 0.2$



(d) $\lambda = 0.5$



(e) $\lambda = 1$

Figure 4.5.6: Simulation results: in each plot, $\hat{\beta}_1$ vs $\hat{\beta}_0$ for true $\lambda = -0.2, 0, 0.2, 0.5, 1$ (from top to bottom).

In Figure 4.5.6, we use scatterplots to identify patterns that result from the correlation between $\hat{\beta}_0$ and $\hat{\beta}_1$ given $\hat{\lambda}_1$, $\hat{\lambda}_2$, $\hat{\lambda}_3$, $\hat{\lambda}_4$ and $\hat{\lambda}_5$, respectively. They provide insights for why the variabilities and outliers in the boxplots in Figure 4.5.2 exist. The two coefficients have a positive association because as $\hat{\beta}_1$ increases, so does $\hat{\beta}_0$. The relationships between $\hat{\beta}_0$ and $\hat{\beta}_1$ become stronger as λ becomes smaller. A considerable amount of variability exists in the estimates of β obtained for $\lambda > 0$. In contrast, for $\lambda = 0$ or close to zero, there are less variabilities in the estimates of β but more outliers.

4.6 Application

Example 4.6.1. the UCBAdmissions data

The `UCBAdmissions` data (R Core Team, 2016) involves applications to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex. The data is in a 3–dimensional array (2×2×6) that is `Admit` (Admitted/Rejected) × `Gender` (Male/Female) × `Dept` (A, B, C, D, E, F). We adopt the way of creating a data frame from this multi-way table `UCBAdmissions` in Maindonald and Braun (2006, p. 258).

R Note:

Import the `UCBAdmissions` data into R, then:

```
UCB <- as.data.frame.table(UCBAdmissions["Admitted", , ])
```

```
names(UCB)[3] <- "admit"

UCB$reject <- as.data.frame.table(UCBAdmissions["Rejected", ,

 ])$Freq

UCB$Gender <- relevel(UCB$Gender, ref="Male")

## Add further columns total and p (proportion admitted)

UCB$total <- UCB$admit + UCB$reject

UCB$p <- cbind(UCB$admit,UCB$total-UCB$admit)
```

The logistic model (or logit model) can be fitted by either the `"logit"` link or our `boxcoxpower(Lambda)` link with `Lambda=0`. The code and summary outputs for the two link functions are:

R Note:

```
library(npmlreg)


model1 <- alldist(p ~  Dept+ Gender, data = UCB, k=1, family=

binomial(link="logit"))


summary(model1)

# Call:  alldist(formula = p ~ Dept + Gender, family =

binomial(link = "logit"),      data = UCB, k = 1)

#

# Coefficients:

#   Estimate Std. Error     t value
```

```
# MASS1          0.58205140 0.06899260   8.4364326

# DeptB         -0.04339793 0.10983890  -0.3951053

# DeptC         -1.26259802 0.10663289 -11.8406063

# DeptD         -1.29460647 0.10582342 -12.2336476

# DeptE         -1.73930574 0.12611350 -13.7915909

# DeptF         -3.30648006 0.16998181 -19.4519642

# GenderFemale  0.09987009 0.08084647   1.2353055

#

# Mixture proportions:

#   MASS1

# 1

#

# Random effect distribution - standard deviation:    0

#

# -2 log L:     89.1     Convergence at iteration  0
```

```
library(boxcoxmix)



model2<-alldist(p ~  Dept+ Gender, data = UCB, k=1, family=

binomial(link=boxcoxpower(0)))



summary(model2)

# Call:  alldist(formula = p ~ Dept + Gender, family =

 binomial(link = boxcoxpower(0)),      data = UCB, k = 1)
```

```
#
# Coefficients:
#   Estimate Std. Error     t value
# MASS1         0.58205140 0.06899260   8.4364326
# DeptB        -0.04339793 0.10983890  -0.3951053
# DeptC        -1.26259802 0.10663289 -11.8406063
# DeptD        -1.29460647 0.10582342 -12.2336476
# DeptE        -1.73930574 0.12611350 -13.7915909
# DeptF        -3.30648006 0.16998181 -19.4519642
# GenderFemale  0.09987009 0.08084647   1.2353055
#
# Mixture proportions:
#   MASS1
# 1
#
# Random effect distribution - standard deviation:    0
#
# -2 log L:     89.1     Convergence at iteration  0
```

As shown above, the results of the two link functions used in fitting our model are identical when λ=0. It means that our approach includes the logistic model as well as the Box-Cox-type models and that allows us to select the best of these models. To find the obtimal λ that maximizes the profile log–likelihood we use the function `boxcoxtype()` as follows

R Note:

```
optim.model <- boxcoxtype(p ~  Dept+ Gender, data = UCB, k=1,
 s=100,find.in.range = c(-3.75,3))
#Maximum Profile Log-likelihood: -36.08529 at lambda= -3.75


summary(optim.model$fit)


Call:  alldist(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
 k = k, random.distribution = random.distribution,
 weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:
                  Estimate   Std. Error    t value
MASS1         2.291159e-01 8.907721e-03 25.7210488
DeptB        -6.265574e-03 9.878346e-03 -0.6342736
DeptC        -2.673655e+00 6.970282e-01 -3.8357917
DeptD        -3.204026e+00 9.071019e-01 -3.5321561
DeptE        -1.585447e+01 5.673514e+00 -2.7944703
DeptF        -6.074866e+03 3.472572e+03 -1.7493850
GenderFemale  3.670963e-02 8.926909e-03  4.1122443
```

```
Mixture proportions:

MASS1

    1



Random effect distribution - standard deviation:    0



-2 log L:     72.2     Convergence at iteration  0
```

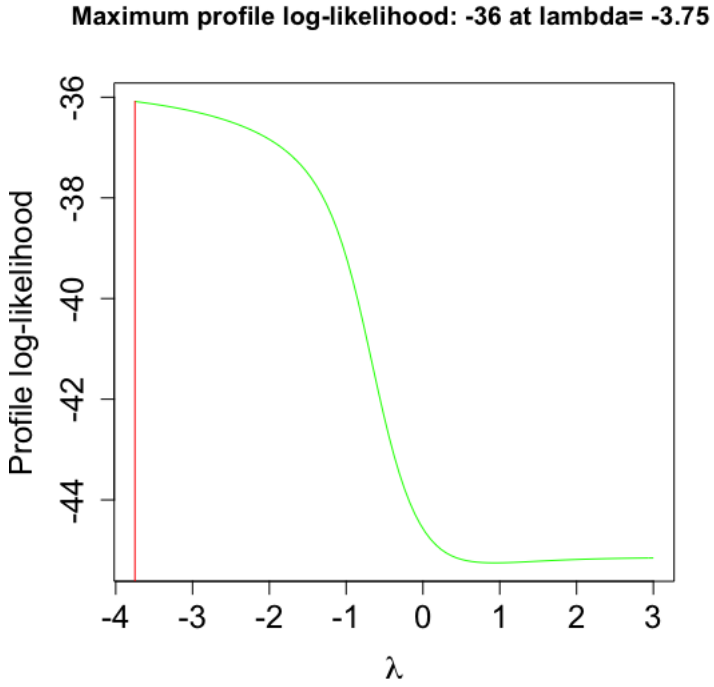


Figure 4.6.1: For the `UCB` data, a grid search over λ

	$\hat{\lambda} = -3.75$	$\lambda = 0$
AIC	90.17059	107.144
BIC	94.53475	151.3138

Table 4.6.1: Comparison of results from logistic & power transformed models for the `UCB` data

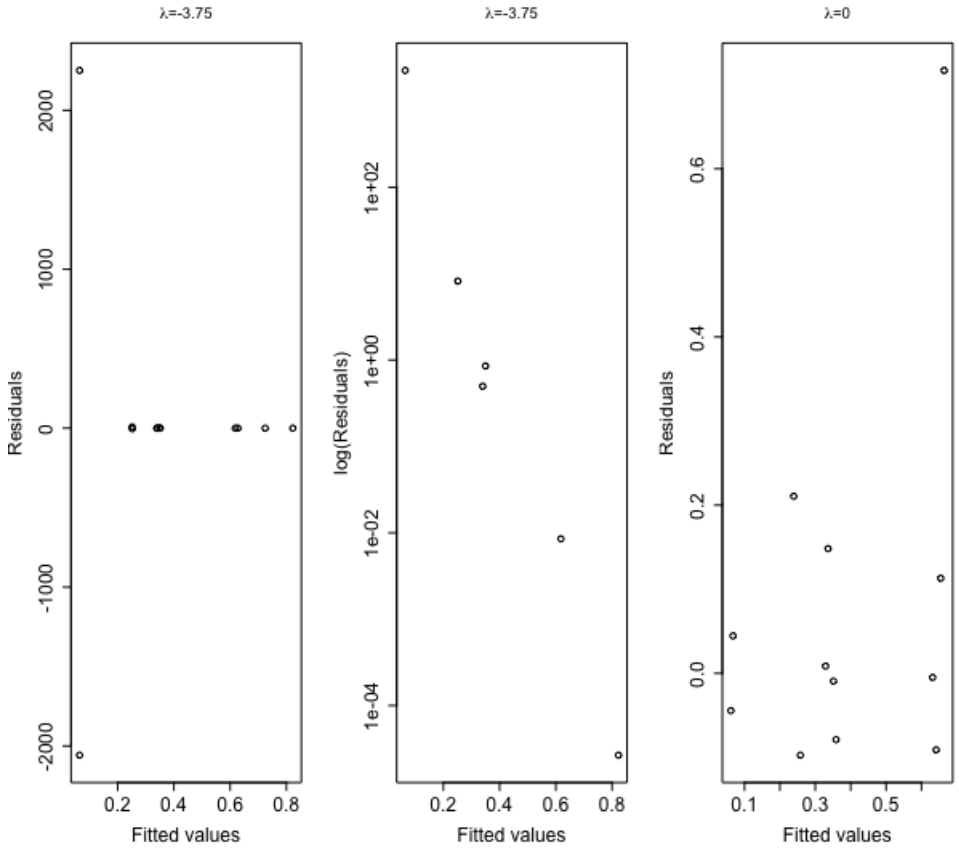


Figure 4.6.2: Residuals against fitted values plots for the `UCB` data using $\hat{\lambda} = -3.75$ and $\lambda = 0$ (logit model). The middle plot is exactly the left plot but with logarithmic scale in the vertical axis.

Figure 4.6.1 shows the plot of the profile likelihood function which summarises information concerning λ, including a vertical line indicating the best estimate of λ. The value $\hat{\lambda}$ is far away from zero, indicating that the log–transformation is not the best choice for this data. Accordingly, the power transformation is suggested with $\hat{\lambda} = -3.75$. Moreover, AIC and BIC criteria defined in (2.6.1) and (2.6.2), are used as model selection criteria, minimum AIC and BIC values are preferred. Table 4.6.1 compares the results from the logit model ($\lambda = 0$) with those of the power transformed model ($\hat{\lambda} = -3.75$). We observed that the power transformed model is significantly better than the logit model in terms of AIC and BIC. However, the -3.75 estimate for λ is at the lower limit of our range, which implies that the

estimation process is trying to send λ to $-\infty$. As one can see in Figure 4.6.1, this is a very strange profile likelihood that quickly grows to infinity as $\lambda \to -\infty$. Similar effects could be observed in the paper of Guerrero and Johnson (1982).

Like the case for the Box-Cox transformed linear mixed model, we cannot compare the estimates of β as they came from completely different models. A very small change of the choice of $\hat{\lambda}$ leads to a considerable change in the estimate of β. However, `DeptF` has very low `admit` probabilities in the logit model and the same happens with the parametric link function. We conduct a residual analysis by plotting the residuals versus the fitted values to detect non-linearity pattern, unequal error variances, and outliers. Figure 4.6.2 shows scatterplots of residuals on the y-axis and fitted values on the x-axis. It is notable that the power link function changes the distribution of the data even though the transformation does not act on the distribution! For $\lambda = -3.75$, the spread of the residuals is decreasing as the fitted values changes, but the local average residual would still be far from 0 with some outliers (extreme values) such as $0/m$ for the binomial distribution. When $\lambda = 0$, the residuals are randomly scattered about zero with a few outliers. This suggests that the assumption that the relationship is linear is more reasonable in the logit model than the power transformed model.

4.7 Discussion

In this Chapter, we tried to offer an alternative to the traditional logit approach using our Box-Cox-type link function for modeling binomial data with the hope

of improving the model fits, but with no success. Our R package **boxcoxmix** has implemented the Box–Cox power transformation of the binary regression model via the function `boxcoxtype()` that operates similarly to the function `optim.boxcox()` for linear models, by creating a profile likelihood and carrying out a grid search over the transformation parameter λ. Also, the **boxcoxmix** package provides a link function `boxcoxpower(Lambda)` that is applicable to the existing R functions `glm()` and `alldist()`, to fit models with fixed value of λ.

We conducted two simulation studies. The first one was based on examining the performance of the `boxcoxpower(Lambda)` link against the `"logit"` link. The related results showed that the latter one was more efficient than the proposed link. However, when λ is close to zero, the proposed link was performing at least as good as the logit model and it was identical to the logit model when $\lambda = 0$. The second study was based on investigating the ability of the proposed method to estimate the transformation and regression parameters simultaneously. The results demonstrated that the proposed method was able to spot the true values of β and λ simultaneously through the function `boxcoxtype()` with more variability as λ gets bigger. However, our simulation studies have fairly large binomial sample sizes ($m_i = 40$) and so this may give approximate normality. It would be interesting to perform simulation with small binomial sample sizes to have better understanding of the performance of proposed approach. In the Example 4.6.1, our method was trying to send λ to $-\infty$ with a very strange profile likelihood that quickly grows to infinity as λ approaches $-\infty$. Similar effects were also observed in the paper of Guerrero and Johnson (1982). This could indicate that there is a potential problem with the proposed approach.

Aranda-Ordaz (1981) used the parametric family of transfomations given by

$$T_\lambda(P) = \frac{2P^\lambda - (1-P)^\lambda}{2P^\lambda + (1-P)^\lambda}$$

where P is the probability of success and λ is the transformation parameter, in order to achieve additivity for binary response data. When $\lambda = 0$, this family reduces to the logistic model and to the linear model when $\lambda = 1$. This family of transformation could be used as an alternative link function as it may be better behaved than the odds-ratio transformations considered here.

Chapter 5

Transformations for mixed–effects logistic models

5.1 Introduction

In this chapter, we focus on binary regression with random effects, following on fixed–effect binary regression models which were presented in Chapter 4. As an alternative to using the log-transformation of the odds-ratio to generalize these models, one may use the Box–Cox transformed link as in the previous Chapter. In this case, we make no assumption about the mixing distribution of the random effects and estimate this distribution using NPML estimation approach as discussed in Chapters 2 and 3. In consideration of the NPML method, Lukociene and Vermunt (2009) studied the performance of the nonparametric and parametric specification of the random effects in multilevel logistic regression models in terms of bias and efficiency. Furthermore,

Lesperance et al. (2014) developed an algorithm that computes the NPML estimates of the mixing distribution of the random effects in the logistic regression model.

This Chapter is organized as follows. Section 5.2 employs the Box–Cox link function discussed in Chapter 4 for mixed–effects binary regression models using the non–parametric profile maximum likelihood technique (NPPML). In a similar way, Section 5.3 uses the link function for the two–level binomial data scenario. Section 5.4 provides a software description of the proposed approach. Results of the simulation study are presented in Section 5.5. Applications to real data sets are given in Section 5.6. Section 5.7 concludes the Chapter.

5.2 Transformations for mixed–effects binary regression models

We now consider the logistic mixed–effects model in which an unobserved random effect z_i, $i = 1, \ldots, n$, with an unspecified distribution $g(z)$ is added to the linear predictor $x_i^T\beta$ for the i-th response, where the response Y_i follows a binomial distribution with success probability P_i and m_i trials, $Y_i \sim B(m_i, P_i)$. The logit form of the logistic model is defined as

$$\log\Big\{P_i/(1-P_i)\Big\} = x_i^T\beta + z_i = \eta_i \qquad , i = 1, ..., n. \tag{5.2.1}$$

where the restrictions $0 < P_i < 1$ and $P_i/(1-P_i) > 0$ apply.

Now the Box–Cox transformation is extended to the generalized linear mixed-effects model. We assume that the power transformation of the odds ratio results in a linear model. That is

$$\Big\{P_i/(1-P_i)\Big\}^{(\lambda)} = x_i^T\beta + z_i = \eta_i \quad , i = 1, ..., n. \tag{5.2.2}$$

The Box–Cox transformation of the odds–ratio is thus,

$$\Big\{P_i/(1-P_i)\Big\}^{(\lambda)} = \begin{cases} \dfrac{\Big\{P_i/(1-P_i)\Big\}^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log\Big\{P_i/(1-P_i)\Big\} & (\lambda = 0) \end{cases} , \tag{5.2.3}$$

where for $\lambda \neq 0$

$$\tilde{P}_i = \Big\{\Big(1+\lambda\eta_i\Big)^{-1/\lambda} + 1\Big\}^{-1} \tag{5.2.4}$$

and

$$1 - \tilde{P}_i = \Big\{\Big(1+\lambda\eta_i\Big)^{1/\lambda} + 1\Big\}^{-1}. \tag{5.2.5}$$

5.2.1 Maximum likelihood estimation of the regression parameters

The conditional probability density function of Y_i is given by

$$f(y_i|\tilde{P}_i) = \binom{m_i}{y_i}\tilde{P}_i^{y_i}(1-\tilde{P}_i)^{m_i-y_i} \tag{5.2.6}$$

where $\tilde{P}_i$ depends on η_i and hence z_i via (5.2.1). The likelihood function is thus

$$L(\lambda, \beta, g) = \prod_{i=1}^{n}\int f(y_i|\tilde{P}_i)g(z_i)dz_i \tag{5.2.7}$$

Recall from our analyses in the previous chapters that the NPML estimate of the (unspecified) mixing distribution $g(z)$ is a discrete distribution involving a finite number K of mass-points z_k, with masses π_k (Aitkin et al., 2009). The likelihood is then

$$L(\lambda, \beta, z_1, ..., z_k, \pi_1, ..., \pi_k) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f(y_i|\tilde{P}_{ik}) \tag{5.2.8}$$

The log-likelihood is then

$$\ell = \log L = \log \Big(\prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k f_{ik}^{(\lambda)} \Big) = \sum_{i=1}^{n} \log \Big(\sum_{k=1}^{K} \pi_k f_{ik}^{(\lambda)} \Big) \tag{5.2.9}$$

where $f_{ik}^{(\lambda)} = f(y_i|\tilde{P}_{ik})$. As in (2.3.7), the "complete data" log-likelihood would be

$$\ell^* = \sum_{i=1}^{n} \sum_{k=1}^{K} \Big[G_{ik} \log \pi_k + G_{ik} \log f_{ik}^{(\lambda)} \Big] \tag{5.2.10}$$

where

$$\begin{aligned} \log f_{ik}^{(\lambda)} &= y_i \log \tilde{P}_{ik} + (m_i - y_i) \log\Big(1 - \tilde{P}_{ik}\Big) \\ &= m_i \log\Big(1 - \tilde{P}_{ik}\Big) + y_i \log \Big\{ \tilde{P}_{ik}/(1 - \tilde{P}_{ik}) \Big\} \\ &= -m_i \log \Big(\Big(1 + \lambda\eta_{ik}\Big)^{1/\lambda} + 1 \Big) + \frac{y_i}{\lambda} \log \Big(1 + \lambda\eta_{ik}\Big) \end{aligned} \tag{5.2.11}$$

then

$$\begin{aligned} \ell^* = \log L^* = \sum_{i=1}^{n} \sum_{k=1}^{K} \Bigg[G_{ik} \log \pi_k + G_{ik} \Big(- m_i \log \Big(\Big(1 + \lambda\eta_{ik}\Big)^{1/\lambda} + 1 \Big) \\ + \frac{y_i}{\lambda} \log \Big(1 + \lambda\eta_{ik}\Big) \Big) \Bigg] \end{aligned} \tag{5.2.12}$$

As in the linear case in Chapter 2, we apply the EM approach to approximate the MLE of the model parameters:

E–step: This is identical to (2.4.10), but $f_{ik}^{(\lambda)}$ here is as in (5.2.6).

M–step: Calculate $\hat{z}_k^{(\lambda)}, \hat{\beta}^{(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ using current w_{ik},

$$\begin{aligned}
\frac{\partial \ell^*}{\partial z_k} &= \sum_{i=1}^{n} w_{ik} \left[-m_i \frac{\lambda(1/\lambda)\big(1+\lambda\eta_{ik}\big)^{(1/\lambda)-1}}{\big(1+\lambda\eta_{ik}\big)^{1/\lambda}+1} + \frac{y_i}{\lambda}\frac{\lambda}{\big(1+\lambda\eta_{ik}\big)} \right] \\
&= \sum_{i=1}^{n} w_{ik} \left[-m_i \frac{\big(1+\lambda\eta_{ik}\big)^{1-\lambda/\lambda}}{\big(1+\lambda\eta_{ik}\big)^{1/\lambda}+1} + y_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
&= \sum_{i=1}^{n} w_{ik} \left[-m_i \frac{\big(1+\lambda\eta_{ik}\big)^{-1}}{\big(1+\lambda\eta_{ik}\big)^{-1/\lambda}+1} + y_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
&= \sum_{i=1}^{n} w_{ik} \left[-m_i\big(1+\lambda\eta_{ik}\big)^{-1}\Big\{\big(1+\lambda\eta_{ik}\big)^{-1/\lambda}+1\Big\}^{-1} + y_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
\Longrightarrow \frac{\partial \ell^*}{\partial z_k} &= \sum_{i=1}^{n} w_{ik}\big(1+\lambda\eta_{ik}\big)^{-1}\Big[-m_i\tilde{P}_{ik} + y_i \Big] \qquad (5.2.13)
\end{aligned}$$

Similarly

$$\begin{aligned}
\frac{\partial \ell^*}{\partial \beta} &= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} \left[-m_i \frac{\lambda x_i(1/\lambda)\big(1+\lambda\eta_{ik}\big)^{(1/\lambda)-1}}{\big(1+\lambda\eta_{ik}\big)^{1/\lambda}+1} + \frac{y_i}{\lambda}\frac{\lambda x_i}{\big(1+\lambda\eta_{ik}\big)} \right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} \left[-m_i \frac{x_i\big(1+\lambda\eta_{ik}\big)^{1-\lambda/\lambda}}{\big(1+\lambda\eta_{ik}\big)^{1/\lambda}+1} + y_i x_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} \left[-m_i \frac{x_i\big(1+\lambda\eta_{ik}\big)^{-1}}{\big(1+\lambda\eta_{ik}\big)^{-1/\lambda}+1} + y_i x_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} \left[-m_i x_i\big(1+\lambda\eta_{ik}\big)^{-1}\Big\{\big(1+\lambda\eta_{ik}\big)^{-1/\lambda}+1\Big\}^{-1} + y_i x_i\big(1+\lambda\eta_{ik}\big)^{-1} \right] \\
\Longrightarrow \frac{\partial \ell^*}{\partial \beta} &= \sum_{i=1}^{n}\sum_{k=1}^{K} w_{ik} x_i\big(1+\lambda\eta_{ik}\big)^{-1}\Big[-m_i\tilde{P}_{ik} + y_i \Big] \qquad (5.2.14)
\end{aligned}$$

and $\hat{\pi}_k^{(\lambda)}$ is as in equation (2.4.14).

Replacing the results into Equation (5.2.9) we get the non-parametric profile log-likelihood function.

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \hat{f}_{ik}^{(\lambda)} \right]. \tag{5.2.15}$$

The NPPML estimate of λ is therefore given by

$$\hat{\lambda} = arg \max_{\lambda} \ \ell_P(\lambda). \tag{5.2.16}$$

which can be found through a grid search over λ.

5.3 Transformations for the two–level binary regression model

The analysis can also be extended to the two–level structure. Let Y_{ij} denote the binomial response of the lower-level units $j = 1, \ldots, n_i$, belonging to the upper-level clusters $i = 1, \ldots, r$, where $\sum_i^r n_i = n$. So Y_{ij} follows a binomial distribution with $Y_{ij} \sim B(m_{ij}, P_{ij})$ with success probability P_{ij} and m_{ij} trials. In this case, it is assumed that there is a value of λ for which,

$$\left\{ P_{ij}/(1 - P_{ij}) \right\}^{(\lambda)} = x_{ij}^T \beta + z_i = \eta_{ij} \tag{5.3.1}$$

where $0 < P_{ij} < 1$ and $P_{ij}/(1 - P_{ij}) > 0$.

In this section, the Box-Cox transformation is extended to the two–level

logistic regression model as

$$\left\{P_{ij}/(1-P_{ij})\right\}^{(\lambda)} = \begin{cases} \dfrac{\left\{P_{ij}/(1-P_{ij})\right\}^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log\left\{P_{ij}/(1-P_{ij})\right\} & (\lambda = 0) \end{cases} \tag{5.3.2}$$

For $\lambda \neq 0$

$$\tilde{P}_{ij} = \left\{\left(1+\lambda\eta_{ij}\right)^{-1/\lambda} + 1\right\}^{-1} \tag{5.3.3}$$

and

$$1-\tilde{P}_{ij} = \left\{\left(1+\lambda\eta_{ij}\right)^{1/\lambda} + 1\right\}^{-1} \tag{5.3.4}$$

5.3.1 Maximum likelihood estimation of the regression parameters

The conditional probability density function of Y_{ij} is given by

$$f(y_{ij}|\tilde{P}_{ij}) = \binom{m_{ij}}{y_{ij}} \tilde{P}_{ij}^{y_{ij}} (1-\tilde{P}_{ij})^{m_{ij}-y_{ij}} \tag{5.3.5}$$

The likelihood can now be approximated using NPML estimation (Aitkin et al., 2009).

$$L(\lambda, \beta, g) = \prod_{i=1}^{r} \int \left[\prod_{j=1}^{n_i} f(y_{ij}|\tilde{P}_{ij})\right] g(z_i) dz_i \approx \prod_{i=1}^{r} \sum_{k=1}^{K} \pi_k \xi_{ik}^{(\lambda)} \tag{5.3.6}$$

where $\xi_{ik}^{(\lambda)} = \prod_{j=1}^{n_i} f(y_{ij}|\tilde{P}_{ijk})$ where $\tilde{P}_{ijk}$ depends on $\eta_{ijk} = x_{ij}^T\beta + z_k$. The log-likelihood is then

$$\ell = \log L = \log\left(\prod_{i=1}^{n}\sum_{k=1}^{K} \pi_k \xi_{ik}^{(\lambda)}\right) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k \xi_{ik}^{(\lambda)}\right) \tag{5.3.7}$$

Using notation as defined in (2.3.7), the "complete data" log-likelihood would be

$$\ell^* = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[G_{ik}\log\pi_k + G_{ik}\log\xi_{ik}^{(\lambda)}\right] \tag{5.3.8}$$

where

$$\begin{aligned}
\log\xi_{ik}^{(\lambda)} &= \sum_{j=1}^{n_i}\log f(y_{ij}|\tilde{P}_{ijk}) \\
&= \sum_{j=1}^{n_i}\left[y_{ij}\log\tilde{P}_{ijk} + (m_{ij}-y_{ij})\log\Big(1-\tilde{P}_{ijk}\Big)\right] \qquad (5.3.9)\\
&= \sum_{j=1}^{n_i} m_{ij}\log\Big(1-\tilde{P}_{ijk}\Big) + \sum_{j=1}^{n_i} y_{ij}\log\left\{\tilde{P}_{ijk}/(1-\tilde{P}_{ijk})\right\} \\
&= -\sum_{j=1}^{n_i} m_{ij}\log\left(\Big(1+\lambda\eta_{ijk}\Big)^{1/\lambda}+1\right) + \sum_{j=1}^{n_i}\frac{y_{ij}}{\lambda}\log\Big(1+\lambda\eta_{ijk}\Big) \qquad (5.3.10)
\end{aligned}$$

then

$$\ell^* = \log L^* = \sum_{i=1}^{n}\sum_{k=1}^{K}\left[G_{ik}\log\pi_k + G_{ik}\sum_{j=1}^{n_i}\left(-m_{ij}\log\left(\Big(1+\lambda\eta_{ijk}\Big)^{1/\lambda}+1\right) + \frac{y_{ij}}{\lambda}\log\Big(1+\lambda\eta_{ijk}\Big)\right)\right] \tag{5.3.11}$$

As in the linear case in Chapter 3, we apply the EM approach to approximate the MLE of the model parameters:

E–step: This is as before but with $f_{ik}^{(\lambda)}$ replaced by $\xi_{ik}^{(\lambda)}$.

M–step: Calculate $\hat{z}_k^{(\lambda)}, \hat{\beta}^{(\lambda)}$ and $\hat{\pi}_k^{(\lambda)}$ using current w_{ik},

$$\begin{aligned}
\frac{\partial\ell^*}{\partial z_k} &= \sum_{i=1}^{r} w_{ik}\sum_{j=1}^{n_i}\left[-m_{ij}\frac{\lambda(1/\lambda)\Big(1+\lambda\eta_{ijk}\Big)^{(1/\lambda)-1}}{\Big(1+\lambda\eta_{ijk}\Big)^{1/\lambda}+1} + \frac{y_{ij}}{\lambda}\frac{\lambda}{\Big(1+\lambda\eta_{ijk}\Big)}\right] \\
&= \sum_{i=1}^{r} w_{ik}\sum_{j=1}^{n_i}\left[-m_{ij}\frac{\Big(1+\lambda\eta_{ijk}\Big)^{1-\lambda/\lambda}}{\Big(1+\lambda\eta_{ijk}\Big)^{1/\lambda}+1} + y_{ij}\Big(1+\lambda\eta_{ijk}\Big)^{-1}\right]
\end{aligned}$$

$$= \sum_{i=1}^{r} w_{ik} \sum_{j=1}^{n_i} \left[-m_{ij} \frac{\left(1+\lambda\eta_{ijk}\right)^{-1}}{\left(1+\lambda\eta_{ijk}\right)^{-1/\lambda}+1} + y_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1} \right]$$

$$= \sum_{i=1}^{r} w_{ik} \sum_{j=1}^{n_i} \left[-m_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1}\left\{\left(1+\lambda\eta_{ijk}\right)^{-1/\lambda}+1\right\}^{-1} + y_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1} \right]$$

$$\implies \frac{\partial \ell^*}{\partial z_k} = \sum_{i=1}^{r} w_{ik} \sum_{j=1}^{n_i} \left(1+\lambda\eta_{ijk}\right)^{-1} \Big[- m_{ij}\tilde{P}_{ijk} + y_{ij} \Big] \tag{5.3.12}$$

Similarly

$$\frac{\partial \ell^*}{\partial \beta} = \sum_{i=1}^{r} \sum_{k=1}^{K} w_{ik} \sum_{j=1}^{n_i} \left[-m_{ij} \frac{\lambda x_{ij}(1/\lambda)\left(1+\lambda\eta_{ijk}\right)^{(1/\lambda)-1}}{\left(1+\lambda\eta_{ijk}\right)^{1/\lambda}+1} + \frac{y_{ij}}{\lambda} \frac{\lambda x_{ij}}{\left(1+\lambda\eta_{ijk}\right)} \right]$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} w_{ik} \sum_{j=1}^{n_i} \left[-m_{ij} \frac{x_{ij}\left(1+\lambda\eta_{ijk}\right)^{1-\lambda/\lambda}}{\left(1+\lambda\eta_{ijk}\right)^{1/\lambda}+1} + y_{ij}x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1} \right]$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} w_{ik} \sum_{j=1}^{n_i} \left[-m_{ij} \frac{x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1}}{\left(1+\lambda\eta_{ijk}\right)^{-1/\lambda}+1} + y_{ij}x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1} \right]$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} w_{ik} \sum_{j=1}^{n_i} \Big[-m_{ij}x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1}\left\{\left(1+\lambda\eta_{ijk}\right)^{-1/\lambda}+1\right\}^{-1} + y_{ij}x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1}$$

$$\implies \frac{\partial \ell^*}{\partial \beta} = \sum_{i=1}^{r} \sum_{k=1}^{K} w_{ik} \sum_{j=1}^{n_i} x_{ij}\left(1+\lambda\eta_{ijk}\right)^{-1} \Big[- m_{ij}\tilde{P}_{ijk} + y_{ij} \Big] \tag{5.3.13}$$

and $\hat{\pi}_k^{(\lambda)}$ is as in equation (3.3.16).

Replacing the results into Equation (5.3.7) we get the non-parametric profile log-likelihood function.

$$\ell_P(\lambda) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \hat{\pi}_k^{(\lambda)} \hat{\xi}_{ik}^{(\lambda)} \right] \tag{5.3.14}$$

The NPPML estimate of λ is then,

$$\hat{\lambda} = arg \max_{\lambda} \ \ell_P(\lambda). \tag{5.3.15}$$

which can be found through a grid search over λ.

5.4 Software description

The **npmlreg** (Einbeck et al., 2014) function `alldist()` can be used again to fit mixed–effects logistic regression model without transformation by setting `family = binomial(link=logit)` and $k > 1$, see Einbeck and Hinde (2009). We also can use the **npmlreg** function `allvc()` to fit the two–level logistic regression model in the same fashion. The proposed approach can be implemented for fixed λ using the **npmlreg** functions with our link function by setting `family = binomial(link= boxcoxpower(Lambda))` where `Lambda` can be any value. In order to perform a grid search over λ, one can use the **boxcoxmix** function `boxcoxtype()` with $k > 1$. In the following sections, we illustrate the use of these functions with simulated and real datasets.

5.5 Simulation study

We are interested in examining the ability of our approach to estimate the transformation and regression parameters of the mixed–effects binary regression models. Accordingly, we conduct two scenarios by following the steps involved in the simulation studies from the previous chapter.

Simulation Study 1

In this case, we generate 1000 datasets with 100 observations by applying the Box–Cox transformation 'backwards' to the success probabilities P_i for each of four given values λ_ℓ, $\ell = 1, 2, 3, 4$. The structure of the simulation data is described briefly in the following,

$$y_{i\ell} \sim B(40, \tilde{P}_i(\lambda_\ell)), \qquad i = 1, ..., 100, \tag{5.5.1}$$

$$\tilde{P}_i(\lambda_\ell) = \begin{cases} \left\{\left(1 + \lambda_\ell \eta_i\right)^{-1/\lambda_\ell} + 1\right\}^{-1} & (\lambda_\ell \neq 0), \\ \left\{1 + e^{-\eta_i}\right\}^{-1} & (\lambda_\ell = 0) \end{cases} \tag{5.5.2}$$

$$\eta_i = 3x_{1,i} + 0.5x_{2,i} + z_i$$

$$X_1 \sim U(-1, 1),\ X_2 \sim U(-1, 1)$$

$$\lambda_1 = 0,\ \ \lambda_2 = 0.2,\ \ \lambda_3 = 0.5,\ \ \lambda_4 = 1.$$

$$z_i \sim \text{Multinomial}\{1, (z_1, z_2, z_3)|\pi_1, \pi_2, \pi_3\}$$

$$z_k = (5, 35, 60) \text{ with masses } \pi_k = 1/3,\ k = 1, 2, 3.$$

As in Chapter 4, $\tilde{P}_i(\cdot)$ denotes the 'backward' Box–Cox–transformation. In addition, the generated data possess a mixed–effects logistic structure due to the random effect terms z_i. Appendix A provides a detailed description of how data were generated in R. Comparing the results of the proposed power link with the logit link for fitting mixed–effects binary regression model shows that the performance of our models seems to be better than the logistic model. However, the proposed link with $\lambda = 0$ corresponds exactly to the logistic model and their results appear identical (Figure 5.5.1).

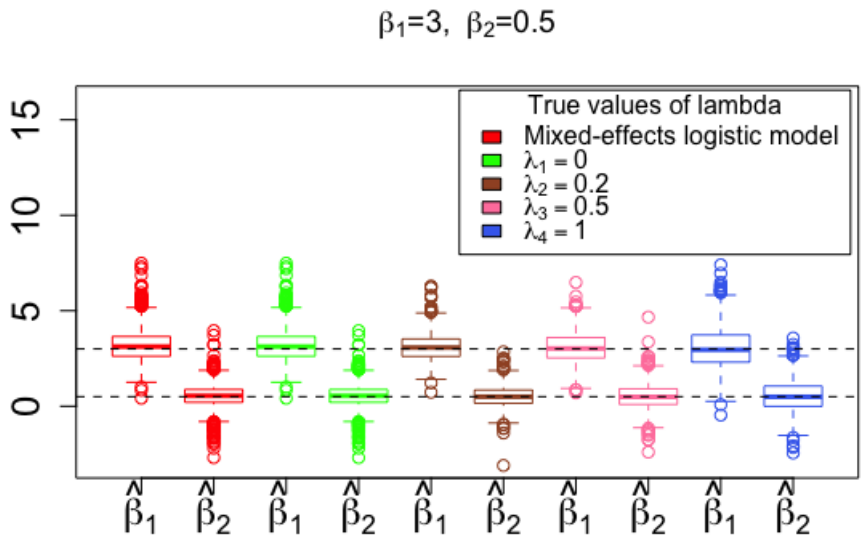


Figure 5.5.1: Simulation results: estimated β for logistic model compared with fixed $\lambda_\ell = 0, 0.2, 0.5, 1$ and $K = 3$ (from left to right), the horizontal lines in the boxplots indicate the actual values of $\beta = 3, 0.5$.

Simulation Study 2

We generated 600 datasets for each value of λ as in the aforementioned study and then estimate the transformation and regression parameters simultaneously through the function `boxcoxtype()`. Figures 5.5.2 and 5.5.3 show the estimates of the regression and transformation parameters, it is clear that the log–transformation (logit model) has some bias. Furthermore, the consistency of the regression parameter estimates becomes stronger as λ increases, and this increases the accuracy of our link compared to the logit link. Unsurprisingly, there are some biases for $\lambda = 0$ or close to zero in terms of estimating the transformation parameters, meaning that the latter biases cause the former. Note that in Figure 5.5.2 (top panel), it is hard to see much here for $\lambda \neq 0$ because of the extreme values for $\lambda = 0$ that are distorting

the scale. Therefore, a cropped version of this plot is provided in the bottom panel.

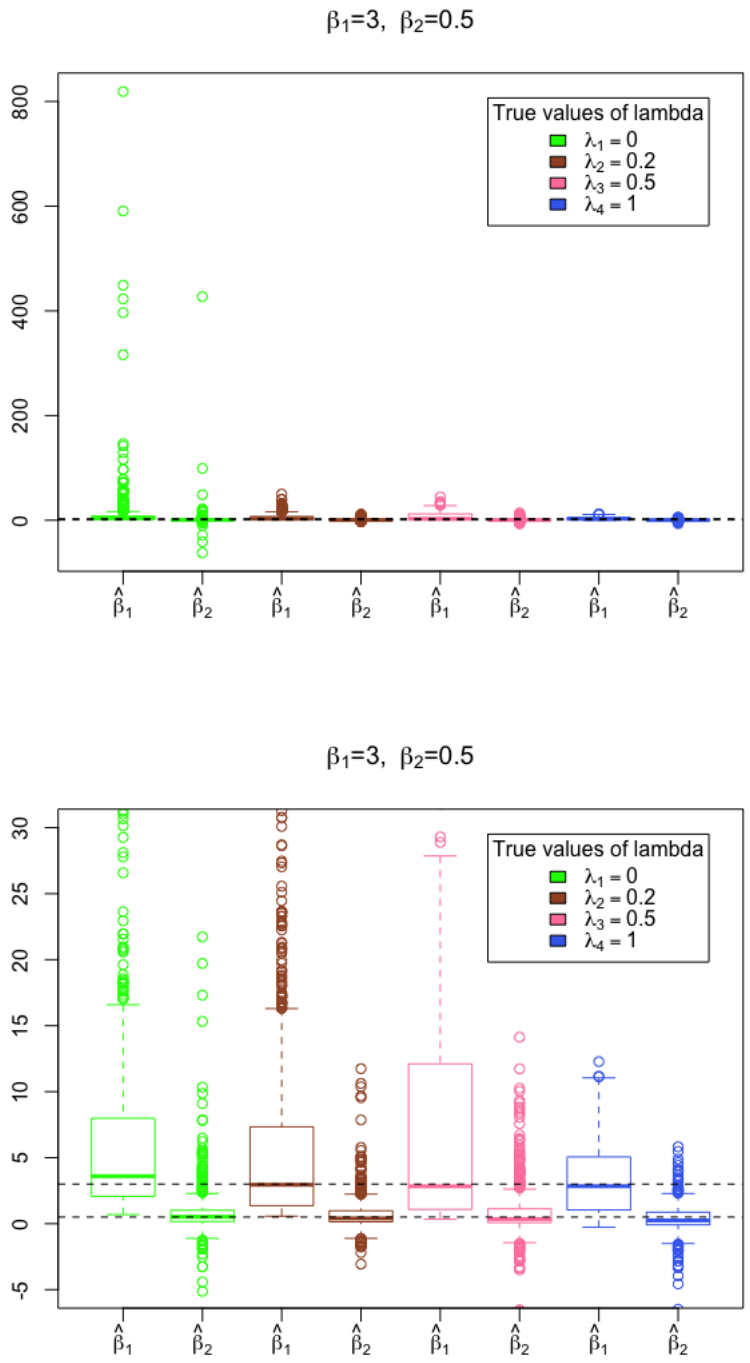


Figure 5.5.2: Simulation results: Estimates $\hat{\beta}$, in each plot for true $\lambda_\ell = 0, 0.2, 0.5, 1$, setting $K = 3$ (from left to right). The lower plot is exactly the upper plot with adjusted limits in the vertical axis in the range of -5 to 30. Horizontal lines indicate the true values $\beta =$ 3, 0.5.

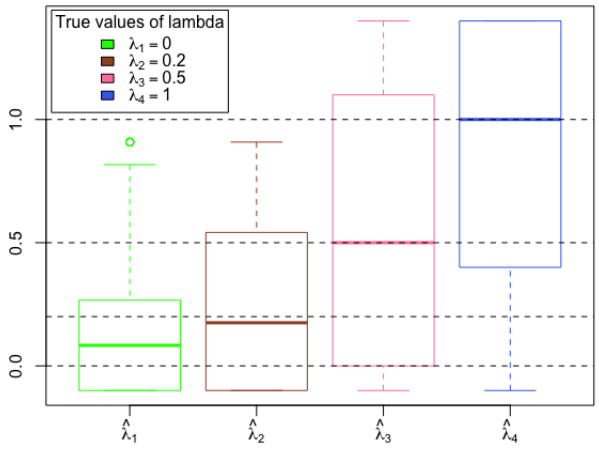
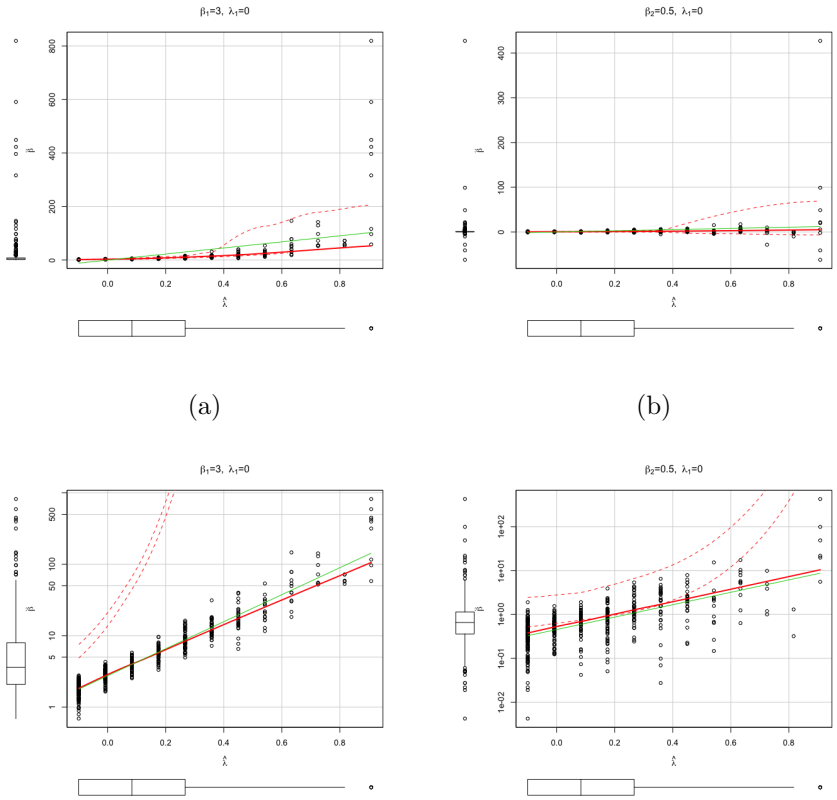


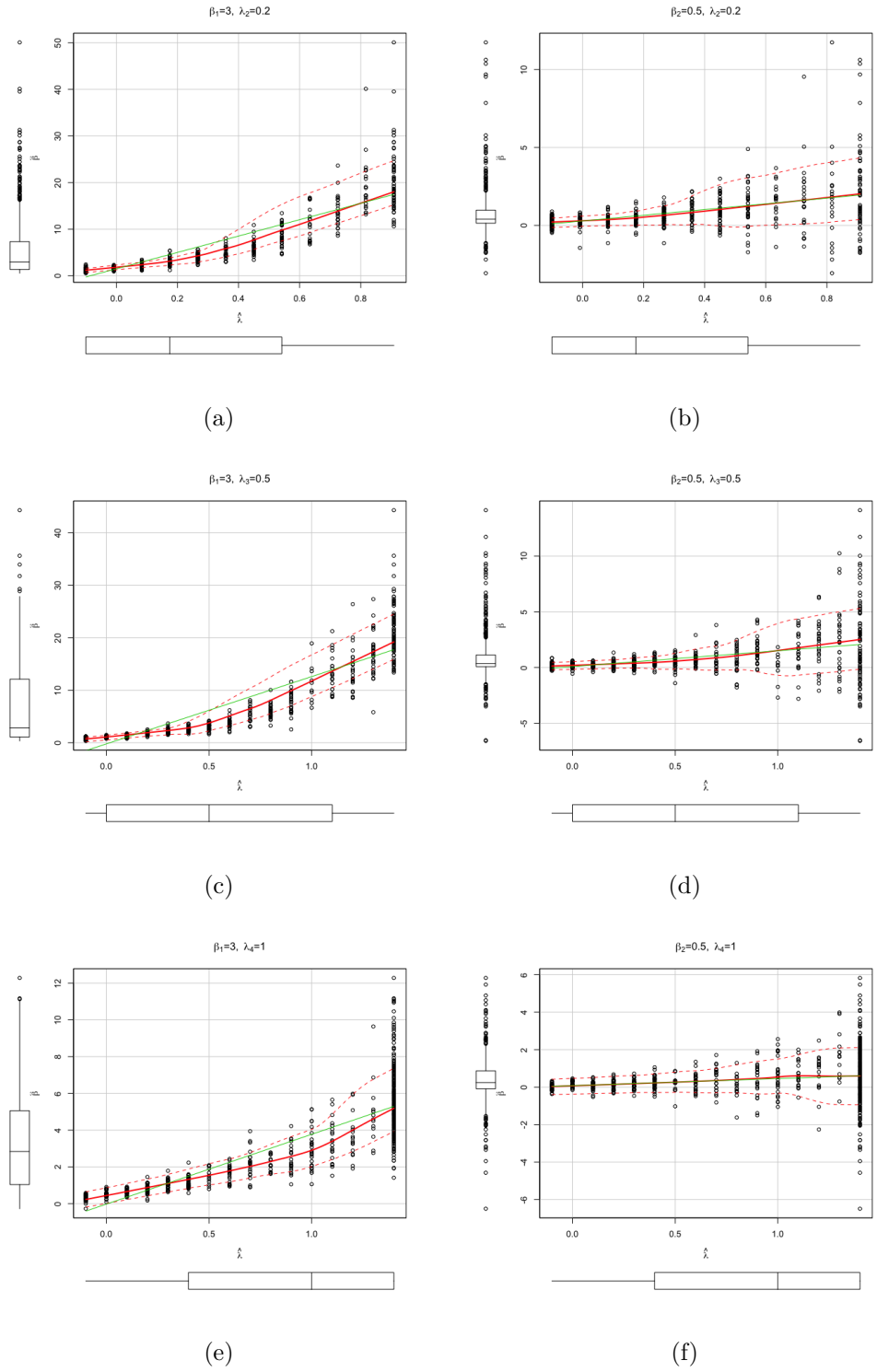
Figure 5.5.3: Simulation results: estimated λ, for true $\lambda_\ell = 0, 0.2, 0.5, 1$, setting $K = 3$ (from left to right). Horizontal lines indicate the true values of λ.



(a) (b)

(c) (d)

Figure 5.5.4: Simulation results: estimated regression parameters against estimated transformation parameters, in each plot for true $\beta_j = 3, 0.5$ (from left to right) and $\lambda_1 = 0$. The lower plots are exactly the upper plots with logarithmic scale in the vertical axes.



(a) (b)

(c) (d)

(e) (f)

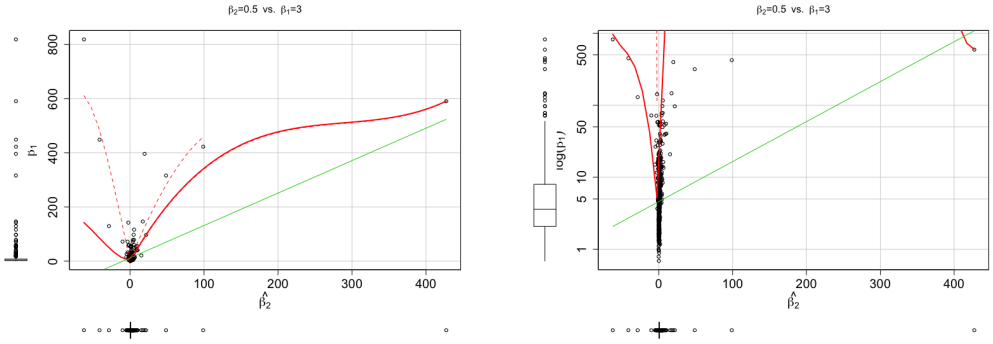
Figure 5.5.5: Simulation results: estimated regression parameters against estimated transformation parameters, in each plot for true $\beta = 3, 0.5$ (from left to right) and $\lambda_\ell = 0.2, 0.5, 1$ (from top to bottom).

The scatterplots of the estimated regression parameters $\hat{\beta}$ against the es-

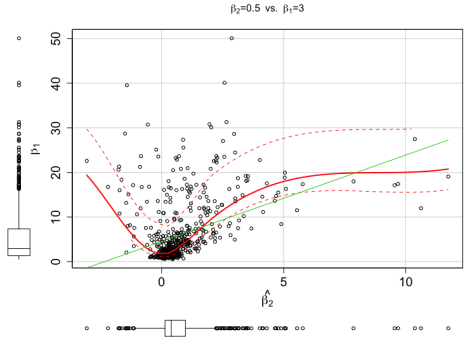
timated transformation parameters $\hat{\lambda}$ for true $\beta = 3, 0.5$ and $\lambda_\ell = 0, 0.2, 0.5, 1$, are shown in Figures 5.5.4 and 5.5.5, respectively. As before, boxplots are added at the margins of each scatterplot to display the position and spread of the estimated parameters and the regression line is used to clarify the trend of the points. The scatterplots suggest that there are curvilinear trends of points, and the variances about the regression lines appear to increase for smaller values of λ. This is unsurprising, as the medians of the estimated transformation parameters for smaller values of λ have some biases, see Figures 5.5.3 and Figure 5.5.4. However, the relationship between the $\hat{\lambda}$ and $\hat{\beta}$ is nearly equivalent to the relationship observed with the Box-Cox transformed fixed–effect binary regression model in Figures 4.5.4 and 4.5.5. In Figure 5.5.6, we use scatterplots to identify patterns that result from the correlation between $\hat{\beta}_1$ and $\hat{\beta}_2$ given $\hat{\lambda}_1$, $\hat{\lambda}_2$, $\hat{\lambda}_3$ and $\hat{\lambda}_4$, respectively. The points patterns in the plots have no direction, the shapes are almost round with some outliers that seem to increase as λ becomes smaller.

An obvious limitation of this simulation is the difficulty of generating data sets for different values of λ given the same starting values. I submitted 5000 jobs to condor queue, each has a single run of simulation but only 600 jobs successfully completed. The rest of jobs experienced an error due to the range of λ that did not work with some simulated data sets; the range of λ needs to be close to the true value of λ that had been used in the simulated data. R codes for the simulation studies are shown in Appendix A. Also, as in the random effect model presented in Chapter 2, the final results are likely sensitive to the design of the simulation. The simulation studies here have binomial sample sizes m_i of 40 which are large relative to the random samples size y_i where $i = 1, \ldots, 100$, this may give approximate

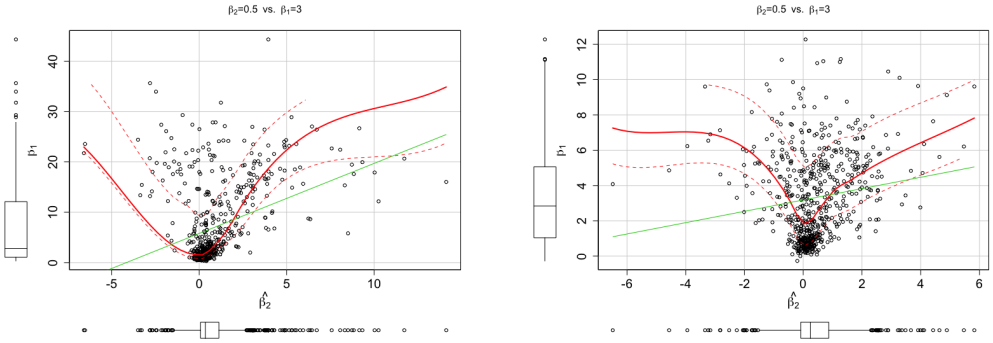
normality. In practice, one can start by defining a range for λ randomly, if any of λ value causes a problem the function `boxcoxtype()` will return an error message that shows which λ value does not work and suggests specifying another range of λ values.



(a) $\lambda = 0$ (b) $\lambda = 0$, $\log(\beta_1)$



(c) $\lambda = 0.2$



(d) $\lambda = 0.5$ (e) $\lambda = 1$

Figure 5.5.6: Simulation results: in each plot, $\hat{\beta}_2$ vs $\hat{\beta}_1$ for true $\lambda = 0, 0.2, 0.5, 1$. Plot (b) is exactly Plot (a) with logarithmic scale in the vertical axis.

5.6 Application

Example 5.6.1. the rainfall data

In order to demonstrate this methodology, we consider the `rainfall` data, also named toxoplasmosis data from the R library **forward** (Scrucca, 2012) which gives the numbers of `Cases` and the `Total` number of observations with a positive test for toxoplasmosis in 34 cities in El Salvador with annual rainfall `Rain` in millimetre. The data have been analyzed in Aitkin et al. (2005) and Einbeck and Hinde (2009) using logistic regression with random effects. We adopt their way of creating the covariates x, x2 and x3, from the variable `Rain`. In this case, we perform a grid search over λ via the proposed power link function that fits logistic–type overdispersion model with two and three mass points separately. Furthermore, AIC and BIC information criteria is used to check model fit, the best–fitting model is the one that minimizes either AIC or BIC with a small number of classes.

R Note:

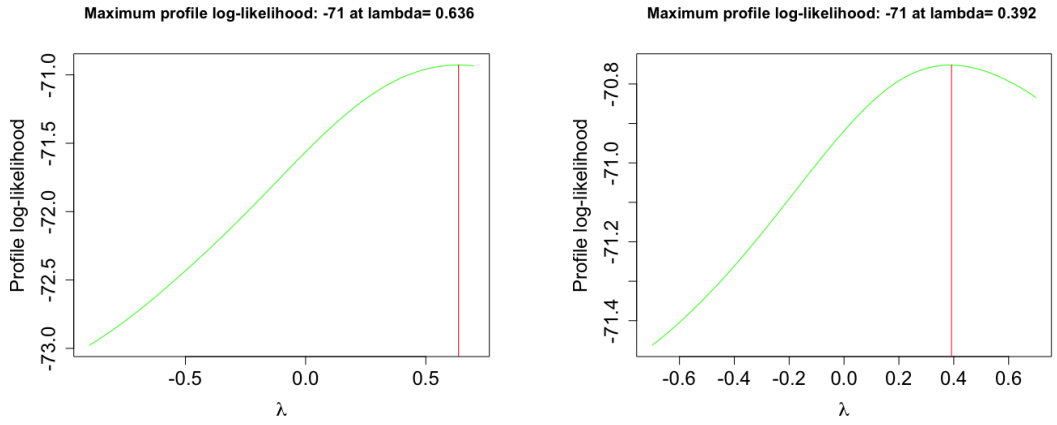
Import the `rainfall` data into R, then:

```
 rainfall$x<-rainfall$Rain/1000
 rainfall$x2<- rainfall$x^2; rainfall$x3<- rainfall$x^3


 library(boxcoxmix)
 model2 <- boxcoxtype(cbind(Cases,Total-Cases) ~ x+x2+x3,
  data = rainfall, k=2, s=100,find.in.range = c(-0.9,0.7))
#Maximum Profile Log-likelihood: -70.92818 at lambda= 0.636
summary(model2$fit)
```

```
Call:  alldist(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
 k = k, random.distribution = random.distribution,
  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:
        Estimate Std. Error   t value
x      567.28081  127.19973  4.459764
x2    -294.83734   65.84321 -4.477870
x3      50.70019   11.30168  4.486075
MASS1 -361.38785   81.48378 -4.435089
MASS2 -360.38376   81.44905 -4.424652


Mixture proportions:
  MASS1    MASS2
0.81303  0.18697


Random effect distribution - standard deviation:    0.391483


-2 log L:     141.9     Convergence at iteration  21
model2$bic
#[1] 166.5409
 model2$aic
```

```
#[1] 155.8564


model3 <- boxcoxtype(cbind(Cases,Total-Cases) ~ x+x2+x3,

data = rainfall, k=3, s=100,find.in.range = c(-0.7,0.7))

#Maximum Profile Log-likelihood: -70.75196 at lambda= 0.392

summary(model3$fit)


Call:  alldist(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

 k = k, random.distribution = random.distribution,

 weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

        Estimate Std. Error   t value

x      576.18347  135.97064  4.237558

x2    -299.93358   70.26897 -4.268364

x3      51.67687   12.04390  4.290710

MASS1 -367.35906   87.27250 -4.209334

MASS2 -366.57687   87.27619 -4.200193

MASS3 -365.66443   87.23292 -4.191817


Mixture proportions:

    MASS1      MASS2      MASS3

0.0935396  0.7291589  0.1773015
```

```
Random effect distribution - standard deviation:    0.4438361


-2 log L:     141.5     Convergence at iteration  22


 model3$bic

#[1] 173.2412

 model3$aic

#[1] 159.5039
```



(a) $K = 2$ (b) $K = 3$

Figure 5.6.1: A grid search over λ, using $K = 2$ (left) and $K = 3$ (right), of the `rainfall` data

Both AIC or BIC select the two–component model as the one that agrees well with the data. The best estimates of λ that maximize $\ell_P(\lambda)$ for the two–component model is $\hat{\lambda} = 0.636$, that is significantly different from zero (Figure 5.6.1(a)). We finally compare the results of our link function `boxcoxpower(Lambda)` using the optimum λ with those obtained from the `logit` link,

R Note:

```
library(npmlreg)
logmodel <- alldist(cbind(Cases,Total-Cases) ~ x+x2+x3,
data = rainfall, k=2, family=binomial(link="logit"))
summary(logmodel)
Call:  alldist(formula = cbind(Cases, Total - Cases) ~ x + x2 + x3
, family = binomial(link = "logit"), data = rainfall, k = 2)


Coefficients:
        Estimate Std. Error   t value
x      592.31583  140.64756  4.211348
x2    -307.70309   72.24154 -4.259365
x3      52.90094   12.30745  4.298285
MASS1 -377.63689   90.86130 -4.156191
MASS2 -376.71432   90.81860 -4.147986


Mixture proportions:
    MASS1      MASS2
0.8269785  0.1730215


Random effect distribution - standard deviation:    0.3489753


-2 log L:     143.1     Convergence at iteration  30
```

```
powermodel <- alldist(cbind(Cases,Total-Cases) ~ x+x2+x3
, data = rainfall, k=2, family=binomial(link=boxcoxpower(0.636)))
summary(powermodel)
Call:  alldist(formula = cbind(Cases, Total - Cases) ~ x + x2 + x3
, family = binomial(link = boxcoxpower(0.636)), data = rainfall,
  k = 2)

Coefficients:
        Estimate Std. Error   t value
x      567.28081  127.19973  4.459764
x2    -294.83734   65.84321 -4.477870
x3      50.70019   11.30168  4.486075
MASS1 -361.38785   81.48378 -4.435089
MASS2 -360.38376   81.44905 -4.424652

Mixture proportions:
  MASS1    MASS2
0.81303  0.18697

Random effect distribution - standard deviation:    0.391483

-2 log L:     141.9     Convergence at iteration  21
```

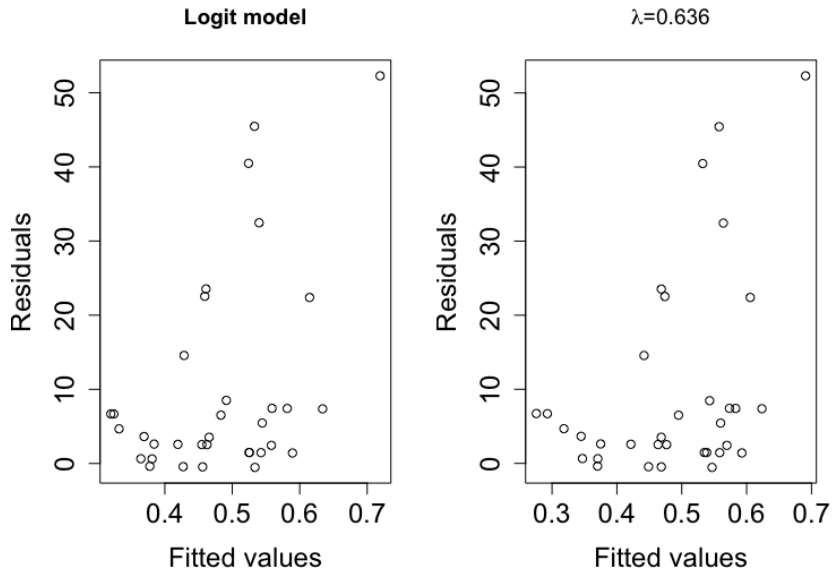


Figure 5.6.2: Residuals against fitted values plots for the `rainfall` data with $K = 2$ using the logit (left plot) and power (right plot) link functions.

link	`logit`	`boxcoxpower(0.636)`
AIC	155.1264	153.8564
BIC	164.2845	163.0145

Table 5.6.1: Comparison of results from logistic & power transformed models for the `rainfall` data

We can see that the proposed power link function performs rather well for this data according to the AIC and BIC comparisons given in Table 5.6.1. Besides, if we look at the residuals against the fitted values plots given in Figure 5.6.2, we can see that our link function does not change the distribution of the data as the transformation acts on the odds instead of the distribution of the data.

Example 5.6.2. the Beta blockers data

The `betablocker` data is a 22–centre clinical trial of β–blockers for reducing mor-

tality after myocardial infarction in patients from the R package **Flexmix** (Leisch, 2004). The data set is analyzed in detail using a finite mixture of binomial logit regressions in Aitkin (1999b) and Grün and Leisch (2007). In the current case, we first use our approach to model overdispersion and then we add the center classification to allow the random effect to have upper–level defined by centers, and lower–level defined by patients. In order to choose the appropriate number of components, a grid search over λ is performed for each component $K \in [2, 5]$, separately, and the AIC and BIC information criteria are used to determine the best model. For this data, the search range for λ is restricted to mainly negative values because the other values of λ beyond this range do not seem to work, computationally. This happens due to the very strong dependence of the proposed link on the sign of the data and λ.

R Note:

Import the `betablocker` data into R, then:

```
beta2 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,
data = betablocker, find.in.range = c(-2,0.4), s=40, k=2,
random.distribution='np')
#Maximum Profile Log-likelihood: -187.8999 at lambda= -0.56


summary(beta2$fit )


Call:  alldist(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
 k = k, random.distribution = random.distribution,
```

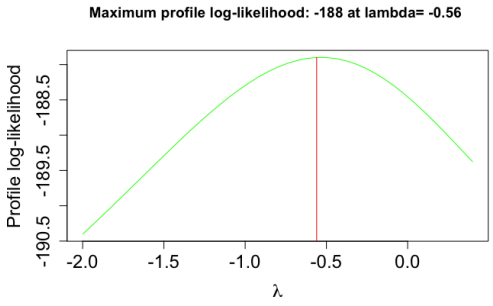
```
 weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                   Estimate Std. Error    t value

TreatmentTreated -0.9857028  0.1733138  -5.687388

MASS1            -5.1692365  0.1450352 -35.641264

MASS2            -2.7001677  0.1297462 -20.811158


Mixture proportions:

    MASS1      MASS2

0.6448312  0.3551688


Random effect distribution - standard deviation:    1.181609


-2 log L:     375.8     Convergence at iteration  9


beta3 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,

 data = betablocker, find.in.range = c(-2,0.4), s=40, k=3,

 random.distribution='np')

#Maximum Profile Log-likelihood: -168.3984 at lambda= 0.4


summary(beta3$fit )
```

```
Call:  alldist(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
  k = k, random.distribution = random.distribution,
  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:
                    Estimate Std. Error    t value
TreatmentTreated -0.08917667 0.02011624  -4.433068
MASS1            -1.72997669 0.02556276 -67.675668
MASS2            -1.48598955 0.01663915 -89.306842
MASS3            -1.16620838 0.02703896 -43.130664


Mixture proportions:
    MASS1      MASS2      MASS3
0.2032775  0.5547424  0.2419801


Random effect distribution - standard deviation:    0.1899316


-2 log L:     336.8     Convergence at iteration  8


beta4 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,
data = betablocker, find.in.range = c(-2,0.3),
 s=40, k=4, random.distribution='np')
```

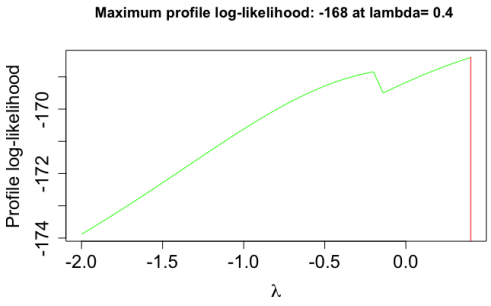
```
# Maximum Profile Log-likelihood: -165.155 at lambda= 0.3


summary(beta4$fit )


Call:  alldist(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                   Estimate Std. Error    t value

TreatmentTreated -0.1305365 0.02575462  -5.068468

MASS1            -1.9518379 0.03453112 -56.524021

MASS2            -1.6573206 0.02135073 -77.623616

MASS3            -1.2384186 0.03265475 -37.924612

MASS4            -1.4818923 0.04058244 -36.515606


Mixture proportions:

    MASS1      MASS2      MASS3      MASS4

0.2006178  0.4361638  0.2291456  0.1340728


Random effect distribution - standard deviation:    0.2410122
```

```
-2 log L:     330.3     Convergence at iteration  28


beta5 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,

data = betablocker, find.in.range = c(-1,0.3),

s=40, k=5, random.distribution='np')

#Maximum Profile Log-likelihood: -165.3097 at lambda= 0.235


summary(beta5$fit )


Call:  alldist(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                   Estimate Std. Error     t value

TreatmentTreated -0.1512355 0.03000781  -5.0398701

MASS1            -2.1233508 5.31711641  -0.3993425

MASS2            -2.1233499 0.04192477 -50.6466640

MASS3            -1.7727755 0.02478336 -71.5308826

MASS4            -1.2963667 0.03634424 -35.6691097

MASS5            -1.5690090 0.04675815 -33.5558411
```

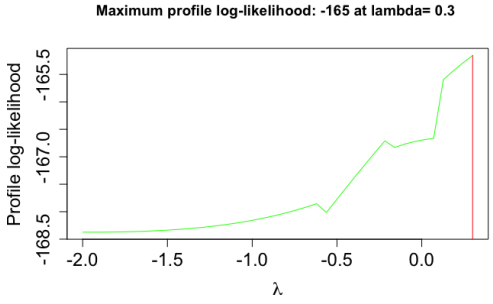
```
Mixture proportions:

       MASS1          MASS2          MASS3          MASS4

1.180478e-05   2.008938e-01   4.362146e-01   2.294046e-01

 MASS5

 1.334752e-01


Random effect distribution - standard deviation:    0.2791726


-2 log L:     330.6     Convergence at iteration  25
```



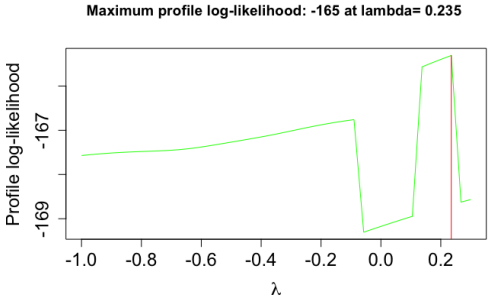
(a) $K = 2$



(b) $K = 3$



(c) $K = 4$



(d) $K = 5$

Figure 5.6.3: A grid search over λ, using $K = 2, 3, 4$ and 5, of the `betablocker` data

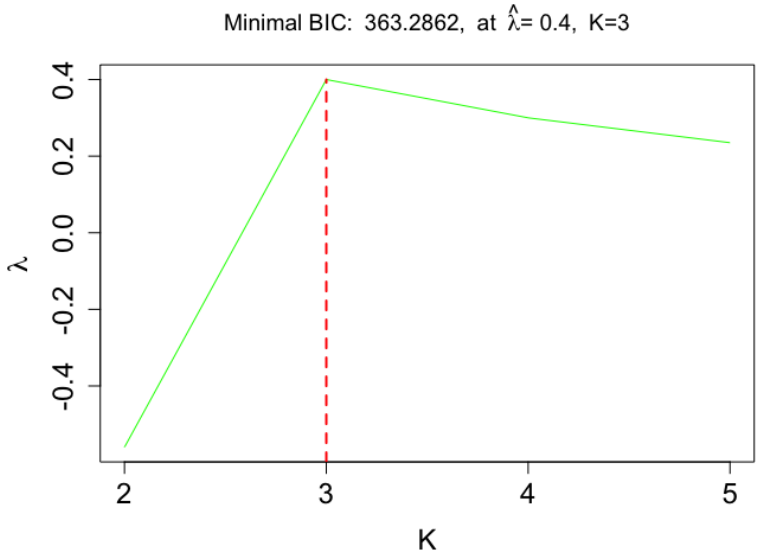


Figure 5.6.4: $\hat{\lambda}$ as a function of K with the optimal `tol` for each class of the `betablocker` data

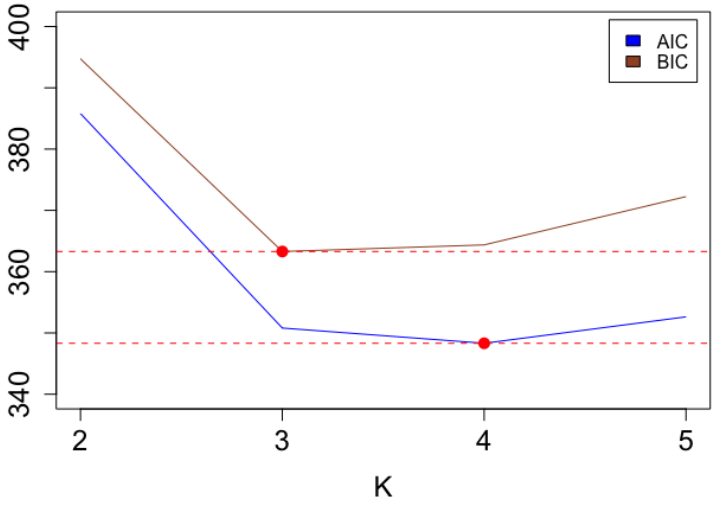


Figure 5.6.5: AIC and BIC values of the Box–Cox–type models for $K \in [2, 5]$ of the `betablocker` data

The best value of λ that maximizes the non–parametric profile log-likelihood of the fitted for each class separately is shown in Figure 5.6.3, while Figure 5.6.4 plots $\hat{\lambda}$ as a function of K with the optimal `tol` for each K. It can be seen that the best

estimates of λ that maximize $\ell_P(\lambda)$ are $\hat{\lambda} = -0.56$, 0.4, 0.3 and 0.235, respectively, suggesting transformations other than log (logit model). Figure 5.6.5 shows the AIC and BIC values of the Box–Cox–type models of the `betablocker` data. In this case a model with three components with $\hat{\lambda} = 0.4$ is preferred according to the BIC. We now analyze this data under the two–level structure of the response,

R Note:

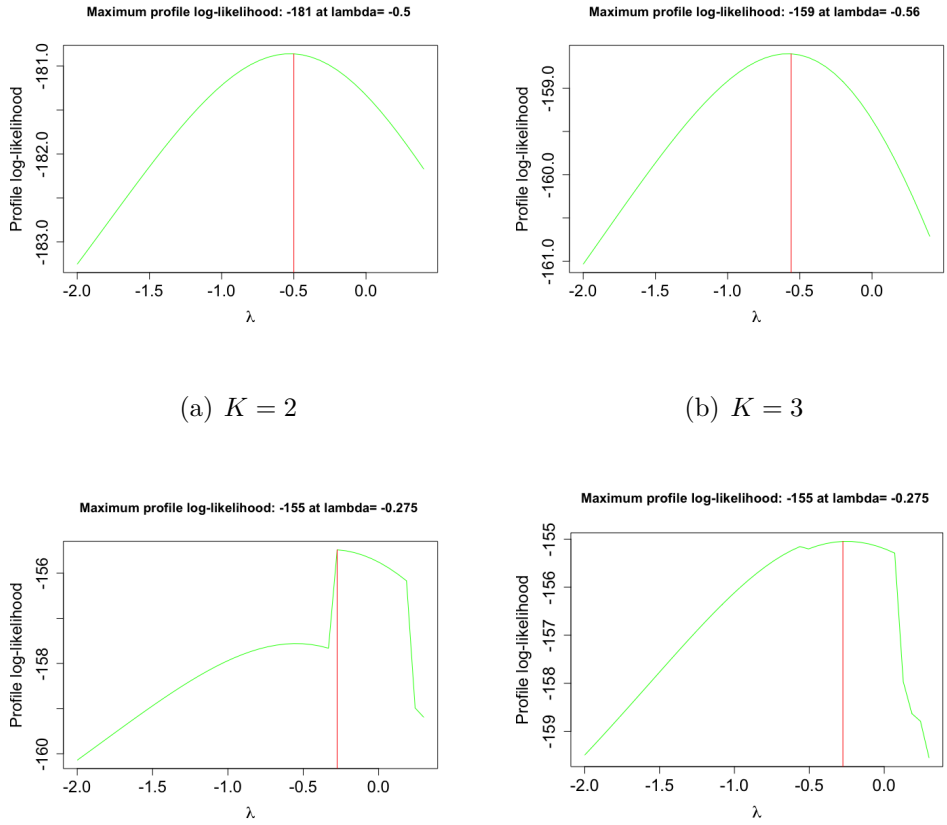
```
betavc2 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,
random=~1|Center, data = betablocker, find.in.range = c(-2,0.4),
s=40,k=2,random.distribution='np')
#Maximum Profile Log-likelihood: -180.8621 at lambda= -0.5


summary(betavc2$fit )


Call:  allvc(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
 k = k, random.distribution = random.distribution,
  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:
                   Estimate Std. Error    t value
TreatmentTreated -0.7872692  0.1523583  -5.167221
MASS1            -4.6687894  0.1227162 -38.045411
```

```
MASS2            -2.4942646  0.1193573 -20.897456


Mixture proportions:

   MASS1     MASS2

0.698617  0.301383


Random effect distribution - standard deviation:    0.9977996


-2 log L:     361.7     Convergence at iteration  9


betavc3 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,

random=~1|Center, data = betablocker, find.in.range = c(-2,0.4),

 s=40, k=3,random.distribution='np')

#Maximum Profile Log-likelihood: -158.6025 at lambda= -0.56


summary(betavc3$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                  Estimate Std. Error    t value
```

```
TreatmentTreated -0.9032108  0.1716431  -5.262146

MASS1            -7.1485025  0.3739917 -19.114066

MASS2            -4.5399919  0.1404007 -32.335952

MASS3            -2.5254135  0.1343205 -18.801398


Mixture proportions:

    MASS1      MASS2      MASS3

0.2392820  0.5116901  0.2490279


Random effect distribution - standard deviation:    1.619826


-2 log L:     317.2     Convergence at iteration  5


betavc4 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,

random=~1|Center, data = betablocker, find.in.range = c(-2,0.3),

s=40, k=4,random.distribution='np')

#Maximum Profile Log-likelihood: -155.4829 at lambda= -0.275


summary(betavc4$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,
```

```
   weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                   Estimate Std. Error    t value

TreatmentTreated -0.4805054 0.09204964  -5.220068

MASS1            -4.3389597 0.16304392 -26.612215

MASS2            -3.1364710 0.07490597 -41.872110

MASS3            -2.2800572 0.11050737 -20.632627

MASS4            -1.7356582 0.10690538 -16.235461


Mixture proportions:

     MASS1       MASS2       MASS3       MASS4

0.23967915  0.48299445  0.09816545  0.17916095


Random effect distribution - standard deviation:    0.876322


-2 log L:     311     Convergence at iteration  14


betavc5 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,

random=~1|Center, data = betablocker, find.in.range = c(-2,0.3),

s=40, k=5,random.distribution='np')

#Maximum Profile Log-likelihood: -155.0474 at lambda= -0.275
```

```
summary(betavc5$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

                   Estimate Std. Error    t value

TreatmentTreated -0.4775353 0.09201865  -5.189549

MASS1            -4.6510676 0.24433044 -19.035973

MASS2            -4.0425405 0.21141213 -19.121611

MASS3            -3.1368324 0.07489620 -41.882399

MASS4            -2.2790565 0.11076010 -20.576511

MASS5            -1.7368351 0.10692608 -16.243326


Mixture proportions:

     MASS1       MASS2       MASS3       MASS4       MASS5

0.16000526  0.07963806  0.48420564  0.09699676  0.17915429


Random effect distribution - standard deviation:    0.9242998


-2 log L:     310.1     Convergence at iteration  13
```



(a) $K = 2$ (b) $K = 3$

(c) $K = 4$ (d) $K = 5$

Figure 5.6.6: A grid search over λ, using $K = 2, 3, 4$ and 5 for the two–level model of the `betablocker` data

As with the overdispersion model, $\hat{\lambda}$'s are different from zero, meaning that logit transformation is not appropriate for this data, see Figures 5.6.6 and 5.6.7. Regarding the suitable number of components, the three mass-points model has the lowest BIC values, indicating that the model with $K = 3$ and $\hat{\lambda} = -0.56$ is the appropriate model for this data (see Figure 5.6.8). Furthermore, the inclusion of the grouping variable (`Center`) in the fitted model with three mass–points reduces the BIC value from 363.2862 to 343.6942.

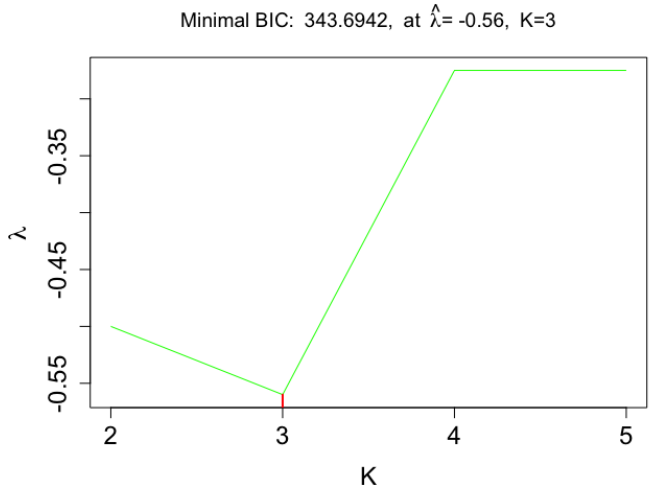


Figure 5.6.7: $\hat{\lambda}$ as a function of K with the optimal `tol` for each class of the two–level model of the `betablocker` data

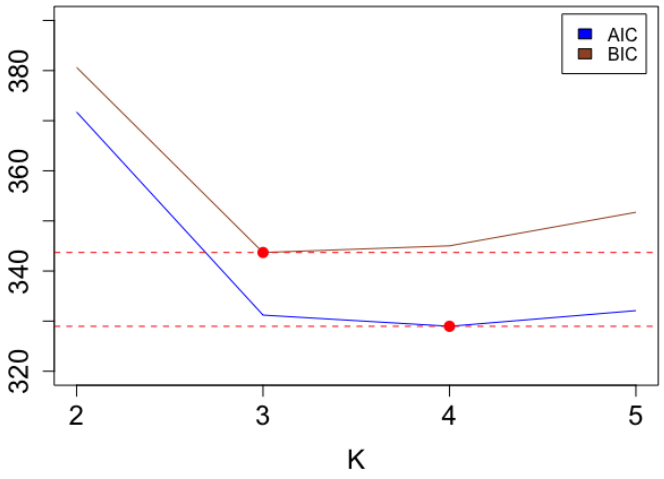


Figure 5.6.8: AIC and BIC values of the Box–Cox–type models for $K \in [2, 5]$ for the two–level model of the `betablocker` data

To see how do conclusions from this model differ from those of other models for this data, we fit the two–level logistic model with $K = 3$.

```
betavcK3logit <-allvc(cbind(Deaths, Total - Deaths) ~ Treatment,
 random=~1|Center, data = betablocker, family = binomial(link =
  logit), k=3,random.distribution='np')
summary(betavcK3logit )


Call:  allvc(formula = cbind(Deaths, Total - Deaths) ~ Treatment,
random = ~1 |Center, family = binomial(link = logit),
data = betablocker, k = 3, random.distribution = "np")


Coefficients:
                  Estimate Std. Error    t value
TreatmentTreated -0.258143 0.04974320  -5.189513
MASS1            -2.833725 0.07368763 -38.455909
MASS2            -2.250088 0.03993349 -56.345898
MASS3            -1.609401 0.05137105 -31.328941


Mixture proportions:
    MASS1      MASS2      MASS3
0.2392002  0.5119035  0.2488963


Random effect distribution - standard deviation:    0.4280793


-2 log L:     318.7     Convergence at iteration  5
```

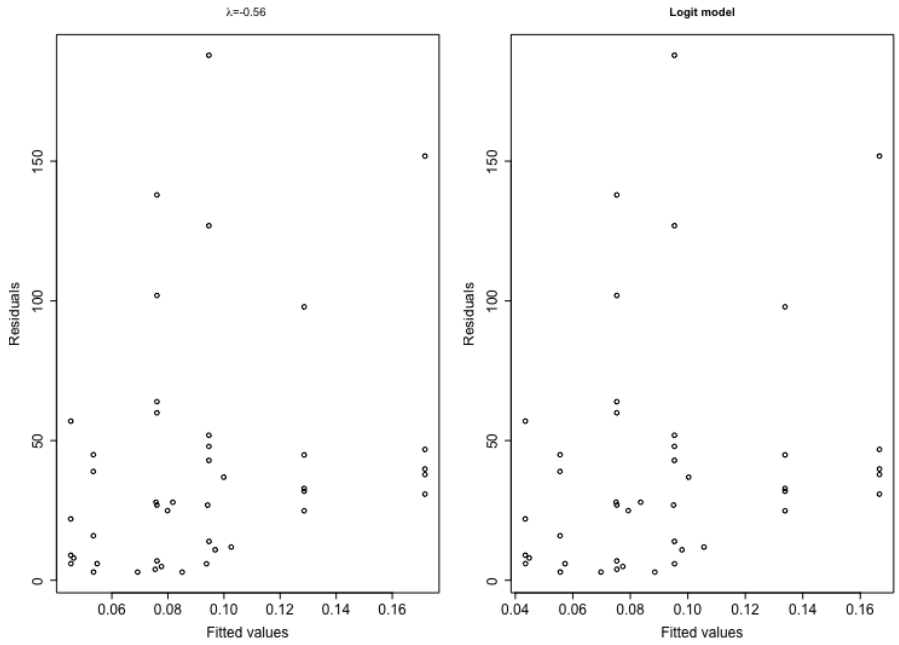


Figure 5.6.9: Residuals against fitted values plots for the two–level model of the `betablocker` data with $K = 3$ using the power (left plot) and logit (right plot) link functions.

Figure 5.6.9 shows the residuals against the fitted values for the two–level model of the `betablocker` data with $K = 3$ using the power and logit link functions. The patterns of the two plots are similar indicating that our link function does not change the distribution of the data.

Example 5.6.3. Mehta Trial data

In this example, we investigate the `Mehta` data (Leisch, 2004) that is a 22-centre clinical trial in a two-level structure whereby each patient that is reported for `Drug`, is nested within one centre (`Site`), where `Drug` indicates treatment with two groups control and receiving a new drug. Note that this data set is similar to the `betablocker` data set analyzed earlier. The data were also studied in Aitkin (1999b) using NPML for logistic regression models with two and three mass–points, and reanalyzed in

Grün and Leisch (2007). In this example, we are interested in comparing the adequacy of the proposed family of power link with the logit link functions in modelling the `Mehta` data.

R Note:

Import the `Mehta` data into R, then:

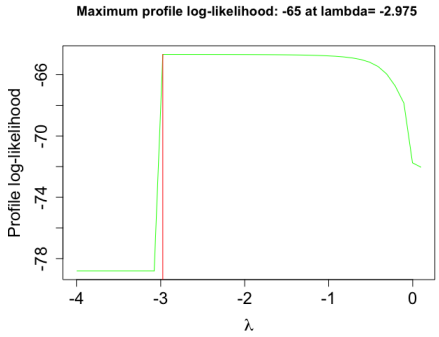
```
Meh2 <-boxcoxtype(cbind(Response, Total - Response)~ Drug,
random=~1|Site, data = Mehta, find.in.range = c(-4,0.1),
s=40, k=2,random.distribution='np')
#Maximum Profile Log-likelihood: -64.67649 at lambda= -2.975

summary(Meh2$fit )

Call:  allvc(formula = formula, random = formula(random),
 family = binomial(link = boxcoxpower(lambda.max)), data = data,
 k = k, random.distribution = random.distribution,
 weights = weights, plot.opt = 0, verbose = FALSE)

Coefficients:
             Estimate Std. Error   t value
DrugControl  21619.11   19045.46  1.135132
MASS1       -21921.85   19044.97 -1.151057
```

```
MASS2       -21618.78   19045.46 -1.135115


Mixture proportions:

     MASS1       MASS2

0.95454535  0.04545465


Random effect distribution - standard deviation:    63.12982


-2 log L:     129.4     Convergence at iteration  31


Meh3 <-boxcoxtype(cbind(Response, Total - Response)~ Drug,

random=~1|Site, data = Mehta, find.in.range = c(-3,0.1),

s=40, k=3,random.distribution='np')

#Maximum Profile Log-likelihood: -62.62554 at lambda= -2.3025


summary(Meh3$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

             Estimate Std. Error    t value
```

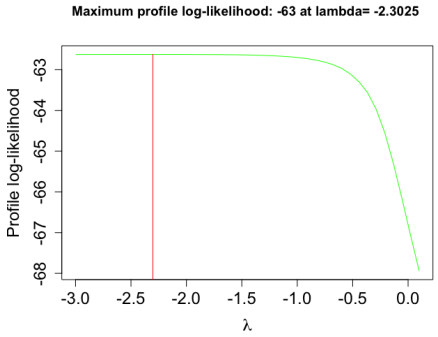
```
DrugControl  1600.710   1186.010  1.3496604

MASS1       -8963.740  10038.516 -0.8929348

MASS2       -1645.052   1185.904 -1.3871708

MASS3       -1600.298   1186.010 -1.3493126


Mixture proportions:

     MASS1       MASS2       MASS3

0.23797093  0.71657445  0.04545462


Random effect distribution - standard deviation:    3117.75


-2 log L:     125.3     Convergence at iteration  18


Meh4 <-boxcoxtype(cbind(Response, Total - Response)~ Drug,

random=~1|Site, data = Mehta, find.in.range = c(-3,0.1),

s=40, k=4,random.distribution='np')

#Maximum Profile Log-likelihood: -62.62573 at lambda= -1.8375


summary(Meh4$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,
```

```
  weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

              Estimate Std. Error   t value

DrugControl   373.2994   226.4685  1.648350

MASS1       -1744.1467  1649.7733 -1.057204

MASS2        -415.1653   226.9457 -1.829360

MASS3        -390.5525   226.4438 -1.724721

MASS4        -372.8052   226.4685 -1.646168


Mixture proportions:

     MASS1       MASS2       MASS3       MASS4

0.21388608  0.27148057  0.46917877  0.04545458


Random effect distribution - standard deviation:    552.0953


-2 log L:     125.3     Convergence at iteration  51


Meh5 <-boxcoxtype(cbind(Response, Total - Response)~ Drug,

random=~1|Site, data = Mehta, find.in.range = c(-4,0),

s=40, k=5,random.distribution='np')

#Maximum Profile Log-likelihood: -62.62772 at lambda= -2.3
```

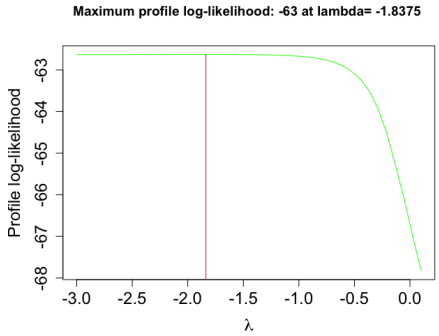
```
summary(Meh5$fit )


Call:  allvc(formula = formula, random = formula(random),

 family = binomial(link = boxcoxpower(lambda.max)), data = data,

  k = k, random.distribution = random.distribution,

   weights = weights, plot.opt = 0, verbose = FALSE)


Coefficients:

              Estimate Std. Error    t value

DrugControl   1596.032   1178.923  1.3538054

MASS1       -15780.070  67276.219 -0.2345564

MASS2        -8890.628  10645.452 -0.8351574

MASS3        -1650.442   1178.874 -1.4000156

MASS4        -1630.093   1178.962 -1.3826511

MASS5        -1595.619   1178.923 -1.3534552


Mixture proportions:

     MASS1       MASS2       MASS3       MASS4       MASS5

0.02468101  0.20706198  0.45030763  0.27249478  0.04545460


Random effect distribution - standard deviation:    3520.92


-2 log L:     125.3     Convergence at iteration  36
```



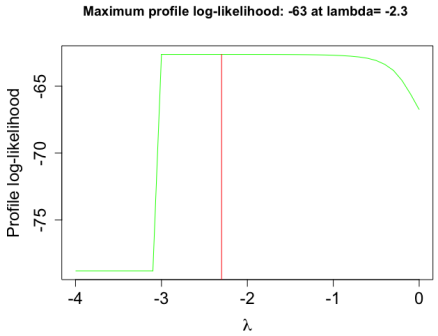
(a) $K = 2$



(b) $K = 3$



(c) $K = 4$



(d) $K = 5$

Figure 5.6.10: A grid search over λ, using $K = 2, 3, 4$ and 5 of the `Mehta` data

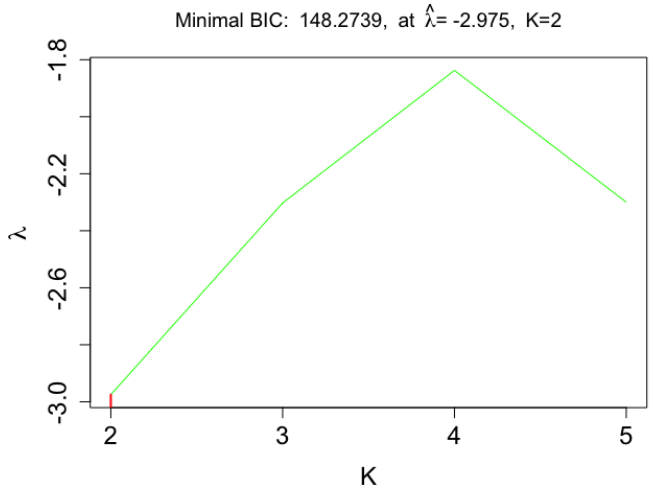


Figure 5.6.11: $\hat{\lambda}$ as a function of K with the optimal `tol` for each class of the `Mehta` data

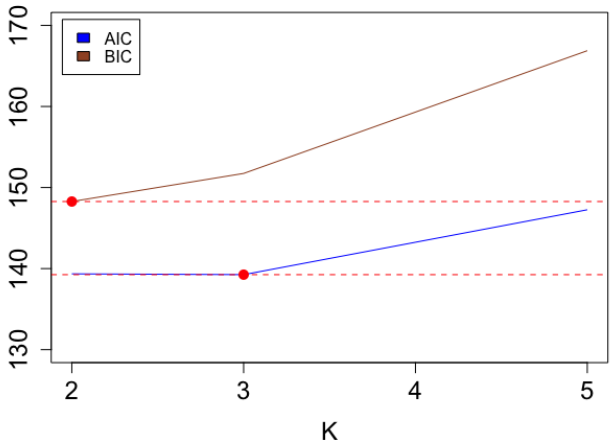


Figure 5.6.12: AIC and BIC values of the Box-Cox-type model for $K \in [2, 5]$ of the `Mehta` data

In Figure 5.6.10, we plot the non–parametric profile log–likelihood values for the fitted model against a set of λ values for each K, $K = 2, 3, 4, 5$, separately. Figure 5.6.11 shows $\hat{\lambda}$ as a function of K with the optimal `tol` for each class. It is clear that $\hat{\lambda}$ differs considerably from 0, for all fitted classes. That provides additional evidence for a better fit of the Box-Cox-type model to the data. We use the AIC and BIC criteria to compare the fitted models, see Figure 5.6.12. The model with two component and $\hat{\lambda} = -2.975$ yields a significantly better fit of the data according to the BIC.

Again, we fit the two level logistic model to with $K = 3$, to compare results from these models for this data.

R Note:

```
Meh2logit <-allvc(cbind(Response, Total - Response)~ Drug,
```

```
random=~1|Site, data = Mehta, family = binomial(link =logit),
 k=2,random.distribution='np')
summary(Meh2logit)


Call:  allvc(formula = cbind(Response, Total - Response) ~
 Drug, random = ~1 |Site, family = binomial(link = logit),
 data = Mehta, k = 2, random.distribution = "np")


Coefficients:
             Estimate Std. Error    t value
DrugControl  1.693484  0.3282503   5.159125
MASS1       -4.147107  0.3169039 -13.086323
MASS2       -2.718306  0.3301630  -8.233225


Mixture proportions:
    MASS1      MASS2
0.8444613  0.1555387


Random effect distribution - standard deviation:    0.5178226


-2 log L:     143.5     Convergence at iteration  30
```

As previously observed for the simpler logistic model with no random effect, the power link function changes the distribution of the data even though the trans-

formation does not act on the distribution! (see Figure 5.6.13). Given this result with the results from the two previous examples, we conclude that if λ is a relatively large negative value then the power link function will change the distribution of the data. For logit model, the spread of the residuals is increasing as the fitted values changes with few outliers. When $\lambda = -2.975$, the residuals in some sense appear to be nonrandom with outliers (extreme values) such as $0/m$ and m/m for the binomial distribution. This suggests that the assumption that the relationship is linear is more reasonable in the logit model than the power transformed model.

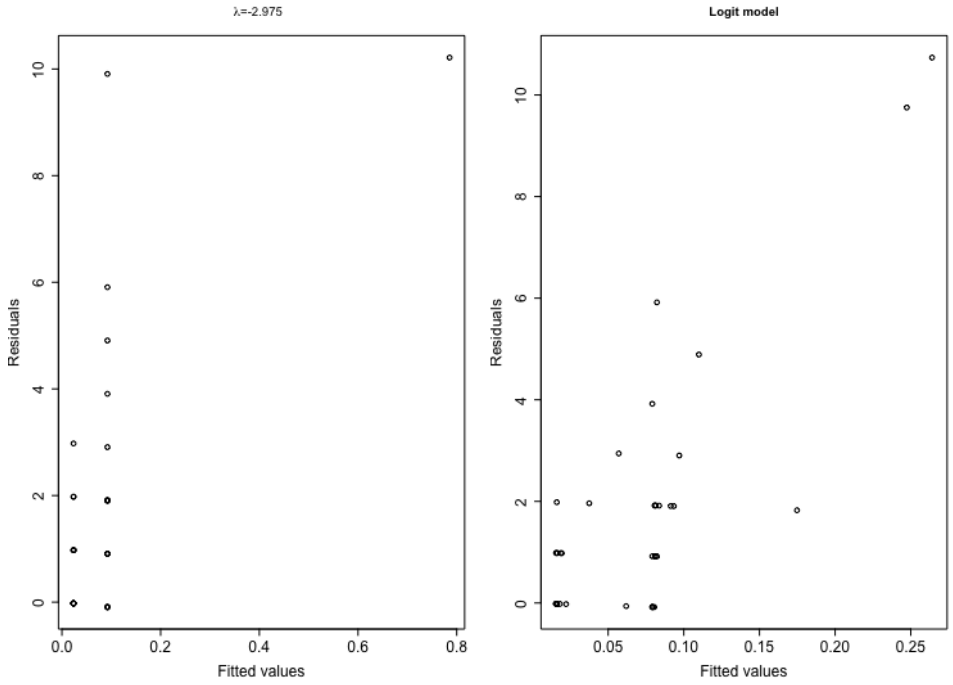


Figure 5.6.13: Residuals against fitted values plots of the `Mehta` data with $K = 2$ using the power (left plot) and logit (right plot) link functions.

5.7 Discussion

In this chapter, we introduced the Box-Cox transformation to the mixed–effects binary regression model as a flexible link function. This family of transformations includes the logit as well as the power transformed models. Simulation results demonstrated that the proposed link is able to spot the true value of λ and hence β for several λ values. However, all of the λ values considered here in the simulation studies are positive, It would be useful to further investigate what would happen for negative values. In some sense, there is some symmetry as for negative values one is simply transforming the odds of the event not happening, than the usual odds of the event occurring for positive λ values. Also, throughout the simulations we used $K = 3$ in the fitting and the simulation. One may try to determine K, this may help to understand the links between K and λ. Furthermore, we have seen that the proposed family link is straightforward to implement and computationally efficient.

From the examples in Section 5.6, we found that all transformed models using $\hat{\lambda}$ that were obtained by the **boxcoxmix** function `boxcoxtype()` gave better fits than the logit transformed model when considering the AIC and BIC criteria or the disparity ($-2\ell_P(\lambda)$), however, by looking at the residuals against the fitted values plots we found that the power link function changed the distribution of the data when λ was a relatively large negative value that led to extreme binomial values such as $0/m$ and m/m. There may be a computational problem that causes difficulty in estimating λ.

Chapter 6

Conclusions and Recommendations

In this thesis, we proposed a transformation approach by extending the Box–Cox transformation to overdispersion and two–level data scenarios in linear models to induce normality and homoscedasticity. We also proposed an alternative link function using the Box–Cox transformation for mixed-effects binary regression models for linearizing purposes. Using the transformation in the presence of random effects with an unspecified mixing distribution can be achieved by using the NPPML technique. To the best of my knowledge, the approach turns out to be the only one of its kind that has implemented the Box–Cox power transformation of the linear and logistic mixed–effects model with unknown random effect distributions.

A number of simulation studies were carried out to evaluate the performance of the proposed methods for linear and logistic mixed–effects models in terms of

bias and efficiency. The results demonstrated that our methods are able to obtain estimates of the transformation and regression parameters simultaneously that are very close to their true values as long as the simulated data is correctly specified. The simulation results were obtained from a misspecified structure of the simulated dataset suggested that the estimation bias for λ causes the estimation bias for β. In the binary regression model framework, the largest number of outliers of the regression parameter estimates found when the true $\lambda = 0$. However, for fixed effect case, a considerable amount of variability exists in the estimates of β obtained for $\lambda > 0$. Surprisingly, the proposed link works better for the binary model with random effects. The potential impact of outliers would be an interesting avenue to explore.

In linear models context, standard errors for the parameter estimates have been proposed and investigated through the simulation data sets, the related results were compared with the robust measure of the standard deviation of the estimated parameters. Although conceptually clear, the EM–based standard errors cannot be 'correct' as they ignore the variation caused by the EM algorithm itself, the simulation results showed that they are satisfyingly close to their empirical counterparts if the variability in the estimates of β was small.

Further simulation analysis to look at model stability would be beneficial to gain a better understanding in the effect of extending the Box–Cox transformation to the linear and logistic models with unknown random effects. All of the λ values considered in the simulation studies were positive, hence, more simulation need to be conducted with negative values of λ before such a conclusion can be arrived at.

In some sense there is some symmetry for binary model as for negative values one is simply transforming the odds of the event not happening, than the usual odds of the event occurring for positive λ values. Also, all of the K values in the simulation studies were fixed, therefore, using K value in the estimation step different from the one that was used in the simulation step can be valuable to explore the interaction between the selection of the transformation and the appropriateness of the random effect.

The question of whether or not to use the Box-Cox transformation for linear and logistic models with unobserved random effects has been answered through real datasets analyses. We are particularly interested in the convergence to normality and homogeneity of variance for the linear model and to improve the fit of the binary regression model. When faced with the decision on whether or not needing to transform the response, not only the value of $\hat{\lambda}$ but also the relevant model selection criteria such as AIC and BIC should be taken into account. It is then essential that these are always based on likelihoods which are reported on the original response scale. Besides, graphical methods were used for measuring normality and homogeneity of variance such as probability plots, control charts, and histograms of residuals. As in the univariate case, the Box-Cox transformation does not guarantee that the assumptions of homoscedasticity and normality of the response distribution in the random effects model is met after applying the transformation, however, it provides a data for which the homoscedasticity and normality assumptions are more reasonable than not applying the transformation at all.

All analyses were conducted in R using the **boxcoxmix** package that is

an implementation of the aforementioned methods. **boxcoxmix** applies the Box-Cox-type transformations to linear and logistic models with random effects using NPML estimation. The function `optim.boxcox()` performs a grid search over the parameter λ for overdispersed and variance component linear models and then optimizes over this grid, to calculate the NPPML estimator of the transformation, while the function `boxcoxtype()` does the same but for binary regression models with fixed effect and mixed-effects with one or two random effect levels. The results allow one to conclude that there is a trade-off between transformation and mixed-effect models, both of them change the nature of the variance explained by the model. Additionally, all transformed models using $\hat{\lambda}$ that were obtained by the **boxcoxmix** functions `optim.boxcox()` and `boxcoxtype()` gave significantly better fits than the untransformed models, when considering the model selection criteria or the disparity. An attempt was unsuccessfully made, in both linear and logistic regression settings, to obtain an easier method that finds the transformation parameter values that maximize the likelihood by deriving the log–likelihood with respect to the transformation parameter. Accordingly, the NPPML estimate of λ that is obtained by plugging in the parameter estimates that were acquired from the EM algorithms tends to be much simpler, straightforward to implement and computationally efficient.

Mixture modeling can also be used to model skewed data, as recognized by Pearson (1895) and McLachlan and Peel (2004). McLachlan and Peel (2004) noted in his book of finite mixture models that "the choice between the log normal and normal mixture model is much interest". As we have seen in Chapter 2, our method can tell us if the data really needs to be transformed or only the right number of components needs to be found in order have a constant variance and

normal distribution. The components of the mixture do not necessarily correspond to clusters of participants within the population (Lubke and Muthén, 2005). In terms of selecting the correct number of classes, Lubke and Muthén (2005) raised the question of whether an extra class can provide a useful information about the heterogeneity. The interplay between normal mixture models and transformations to achieve homogeneous variances deserves considerable attention.

Chapter 2 raised the question of whether the restriction on the response to be greater than zero has an effect on the results of the transformation. Therefore, it would be interesting to apply in a similar fashion the shifted power transformation to the linear and logistic models with random effects. An interesting alternative approach to that for binary models is to consider the Aranda-Ordaz (1981) families of transformations of the probability that may be better behaved than the odds-ratio transformations considered in Chapters 4 and 5. Further research is recommended that extends our approach to generalized linear mixed models with Box–Cox type link functions. Note that this idea was explored extensively in the special case of logistic models in Chapters 4 and 5. It would be also interesting to combine the Box-Cox type link with the standard Box-Cox response transformation for linear mixed models. Moreover, one could easily consider the proposed approaches using gaussian quadrature, where instead of estimating z_k and π_k, one uses fixed values as tabulated by Hinde (1982). Our approaches assumed a single λ for all of the components, alternatively one could use a different λ for each component K, *i.e.* λ_k, it seems plausible that each of these components need to be transformed in a different way.

References

Agresti, A., Caffo, B., and Ohman-Strickland, P. (2004). Examples in which misspecification of a random effects distribution reduces efficiency, and possible remedies. *Computational Statistics & Data Analysis*, 47(3):639–653.

Aitkin, M. (1995). NPML estimation of the mixing distribution in general statistical models with unobserved random effects. *Statistical Modelling*, pages 1–9.

Aitkin, M. (1996a). A General Maximum Likelihood Analysis of Overdispersion in Generalized Linear Models. *Statistics and Computing*, 6(3):251–262.

Aitkin, M. (1996b). Empirical Bayes shrinkage using posterior random effect means from nonparametric maximum likelihood estimation in general random ef- fect models. *Statistical Modelling: Proceedings of the 11th IWSM*, pages 87–94.

Aitkin, M. (1999a). A General Maximum Likelihood Analysis of Variance Components in Generalized Linear Models. *Biometrics*, 55(1):117–128.

Aitkin, M. (1999b). Meta-analysis by random effect modelling in generalized linear models. *Statistics in Medicine*, 18(17-18):2343–2351.

Aitkin, M. A., Francis, B., and Hinde, J. (2005). *Statistical Modelling in GLIM 4*, volume 32. Oxford University Press, USA.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. (2009). *Statistical Modelling in R*. Oxford University Press Oxford.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer.

Almohaimeed, A. and Einbeck, J. (2017). *boxcoxmix: Response Transformations for*

Random Effect and Variance Component Models. R package version 0.13.

Aranda-Ordaz, F. J. (1981). On two families of transformations to additivity for binary response data. *Biometrika*, 68(2):357–363.

Asar, Ö., Ilk, O., and Dag, O. (2017). Estimating Box-Cox power transformation parameter via goodness-of-fit tests. *Communications in Statistics-Simulation and Computation*, 46(1):91–105.

Assaf, A. G., Oh, H., and Tsionas, M. G. (2016). Unobserved heterogeneity in hospitality and tourism research. *Journal of Travel Research*, 55(6):774–788.

Baker, S. G. (1992). A simple method for computing the observed information matrix when using the em algorithm with categorical data. *Journal of Computational and Graphical Statistics*, 1(1):63–76.

Bhat, H. S. and Kumar, N. (2010). On the derivation of the bayesian information criterion. *School of Natural Sciences, University of California*.

Bock, R. D. and Aitkin, M. (1981). Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm. *Psychometrika*, 46(4):443–459.

Bowman, A. and Evers, L. (2017). Nonparametric Smoothing Lecture Notes.

Box, G. E. and Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252.

Butler, S. M. and Louis, T. A. (1992). Random effects models with non-parametric priors. *Statistics in medicine*, 11(14-15):1981–2000.

Böhning, D., Kuhnert, R., Viwatwongkasem, C., and Rattanasiri, S. (2006). Non-parametric Profile Likelihood Estimation in Meta-Analysis with Individually Pooled Data.

Carroll, R. J. (1982). Prediction and power transformations when the choice of power is restricted to a finite set. *Journal of the American Statistical Association*, 77(380):908–915.

Changyong, F., Hongyue, W., Naiji, L., Tian, C., Hua, H., and Ying, L. (2014). Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, 26(2):105.

Claeskens, G. (2016). Statistical model choice.

Clark, T. S. and Linzer, D. A. (2015). Should I use fixed or random effects? *Political Science Research and Methods*, 3(2):399–408.

da Silva-Júnior, A. H. M., da Silva, D. N., and Ferrari, S. L. P. (2014). mdscore: An R Package to Compute Improved Score Tests in Generalized Linear Models. *Journal of Statistical Software*, 61(2):1–16.

Dalgaard, P. (2008). *Introductory statistics with R*. Springer Science & Business Media.

Davies, R. (1987). Mass Point Methods for Dealing with Nuisance Parameters in Longitudinal Studies. *Longitudinal Data Analysis*, pages 88–109.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, pages 1–38.

Einbeck, J., Darnell, R., and Hinde, J. (2014). *npmlreg: Nonparametric Maximum Likelihood Estimation for Random Effect Models*. R package version 0.46-1.

Einbeck, J. and Hinde, J. (2006). A Note on NPML Estimation for Exponential

Family Regression Models with Unspecified Dispersion Parameter. *Austrian Journal of Statistics.*, 35(2&3):233–243.

Einbeck, J. and Hinde, J. (2009). Nonparametric Maximum Likelihood Estimation for Random Effect Models in R. *Vignette to R package npmlreg version 0.44.*

Einbeck, J., Hinde, J., and Darnell, R. (2007). A New Package for Fitting Random Effect Models. *R news.*, 7(1):26–30.

Foster, A., Tian, L., and Wei, L. (2001). Estimation for the Box-Cox transformation model without assuming parametric error distribution. *Journal of the American Statistical Association*, 96(455):1097–1101.

Fotouhi, A. R. (2003). Comparisons of estimation procedures for nonlinear multilevel models. *J Stat Softw*, 8:1–39.

Friedl, H. and Kauermann, G. (2000). Standard errors for em estimates in generalized linear models with random effects. *Biometrics*, 56(3):761–767.

Gray, E. (2016). Constructing league tables using random effect models.

Grün, B. and Leisch, F. (2007). Applications of finite mixtures of regression models. *URL: http://cran. r-project. org/web/packages/flexmix/vignettes/regression-examples. pdf.*

Guerrero, V. M. and Johnson, R. A. (1982). Use of the Box-Cox transformation with binary response models. *Biometrika*, 69(2):309–314.

Gurka, M. J. (2004). The box-cox transformation in the general linear mixed model for longitudinal data.

Gurka, M. J., Edwards, L. J., Muller, K. E., and Kupper, L. L. (2006). Extending

the Box-Cox Transformation to the Linear Mixed Model. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 169(2):273–288.

Heckman, J. and Singer, B. (1984). A Method for Minimizing the Impact of Distributional Assumptions in Econometric Models for Duration Data. *Econometrica: Journal of the Econometric Society*, pages 271–320.

Hinde, J. (1982). Compound poisson regression models. In *GLIM 82: Proceedings of the International Conference on Generalised Linear Models*, pages 109–121. Springer.

Hinde, J. and Demétrio, C. (2007). Overdispersion: Models and Estimation A Short Course for SINAPE 1998. *Statistics*.

Hou, Q., Mahnken, J. D., Gajewski, B. J., and Dunton, N. (2011). The Box-Cox Power Transformation on Nursing Sensitive Indicators: Does it Matter if Structural Effects are Omitted During the Estimation of the Transformation Parameter? *BMC Medical Research Methodology*, 11(1):1.

Ji, Y., Wang, L., Zhang, H., and Zhou, Y. (2017). Semiparametric estimation of a Box-Cox transformation model with varying coefficients model. *Science China Mathematics*, 60(5):897–922.

Karlis, D. and Xekalaki, E. (2003). Choosing Initial Values for the EM Algorithm for Finite Mixtures. *Computational Statistics & Data Analysis*, 41(3):577–590.

Laird, N. (1978). Nonparametric Maximum Likelihood Estimation of a Mixing Distribution. *Journal of the American Statistical Association*, 73(364):805–811.

Leisch, F. (2004). Flexmix: A general framework for finite mixture models and latent glass regression in R.

Leroux, B. G. and Puterman, M. L. (1992). Maximum-penalized-likelihood estim-

ation for independent and Markov-dependent mixture models. *Biometrics*, pages 545–558.

Lesperance, M., Saab, R., and Neuhaus, J. (2014). Nonparametric estimation of the mixing distribution in logistic regression mixed models with random intercepts and slopes. *Computational Statistics & Data Analysis*, 71:211–219.

Lindsay, B. G. (1983). The Geometry of Mixture Likelihoods: a General Theory. *The Annals of Statistics*, 11(1):86–94.

Lindstrom, M. J. and Bates, D. M. (1988). Newton—Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*, 83(404):1014–1022.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 226–233.

Lubke, G. H. and Muthén, B. (2005). Investigating population heterogeneity with factor mixture models. *Psychological methods*, 10(1):21.

Lukociene, O. (2010). *Latent class models for categorical data with a multilevel structure*. Universiteit van Tilburg.

Lukociene, O. and Vermunt, J. K. (2009). Logistic regression analysis with multidimensional random effects: A comparison of three approaches. *Submitted for publication*.

Maindonald, J. and Braun, J. (2006). *Data analysis and graphics using R: an example-based approach*, volume 10. Cambridge University Press.

Maruo, K., Isogawa, N., and Gosho, M. (2015). Inference of median difference

based on the Box–Cox model in randomized clinical trials. *Statistics in medicine*, 34(10):1634–1644.

Maruo, K., Yamaguchi, Y., Noma, H., and Gosho, M. (2017). Interpretable inference on the mixed effect model with the Box–Cox transformation. *Statistics in Medicine*, 36(15):2420–2434.

McLachlan, G. and Peel, D. (2004). *Finite mixture models*. John Wiley & Sons.

Murphy, S. A. and Van der Vaart, A. W. (2000). On profile likelihood. *Journal of the American Statistical Association*, 95(450):449–465.

Nawata, K. (1994). Estimation of Sample Selection Bias Models by the Maximum Likelihood Estimator and Heckman's Two-Step Estimator. *Economics Letters*, 45(1):33–40.

Nawata, K. (2013). A new estimator of the Box-Cox Transformation Model Using Moment Conditions. *Economics Bulletin*, 33(3):2287–2297.

Osborne, J. W. (2010). Improving your data transformations: Applying the Box-Cox transformation. *Practical Assessment, Research & Evaluation*, 15(12):2.

Ostle, B. and Malone, L. C. (1954). Statistics in Research: Basic Concepts and Techniques for Research Workers. Technical report, JSTOR.

Pearson, K. (1895). Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London*, 186(Part I):343–424.

Piepho, H.-P. and McCulloch, C. E. (2004). Transformations in mixed models: Application to risk analysis for a multienvironment trial. *Journal of agricultural, biological, and environmental statistics*, 9(2):123–137.

Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., and R Core Team (2016). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-128.

Polańska, J. (2003). The EM Algorithm and its Implementation for the Estimation of Frequencies of SNP-Haplotypes. *International Journal of Applied Mathematics and Computer Science*, pages 419–429.

Qarmalah, N. M., Einbeck, J., and Coolen, F. P. A. (2018). k-Boxplots for mixture data. *Statistical Papers*, 59(2):513–528.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rabe-Hesketh, S., Pickles, A., and Skrondal, A. (2003). Correcting for covariate measurement error in logistic regression using nonparametric maximum likelihood estimation. *Statistical Modelling*, 3(3):215–232.

Rao, M. B. and Rao, C. R. (2014). *Computational Statistics with R*, volume 32. Elsevier.

Rodriguez, G. (2012). Generalized linear models. *Notes [assessed on 1 May 2010]. available at http://data. princeton. edu/wws509/notes*.

Sakia, R. (1992). The Box-Cox transformation technique: a review. *The statistician*, pages 169–178.

Scrucca, L. (2012). *forward: Forward search*. R package version 1.0.3.

Shin, Y. (2008). Semiparametric estimation of the Box–Cox transformation model. *The Econometrics Journal*, 11(3):517–537.

Shuster, J. and Miura, C. (1972). Two-Way Analysis of Reciprocals. *Biometrika*, 59(2):478–481.

Silverman, B. W. (1986). Density estimation for statistics and data analysis.

Sofroniou, N., Einbeck, J., and Hinde, J. (2006). Analyzing Irish suicide rates with mixture models. National University of Ireland.

Solomon, P. (1985). Transformations for Components of Variance and Covariance. *Biometrika*, 72(2):233–239.

Spitzer, J. J. (1982). A primer on Box-Cox estimation. *The Review of Economics and Statistics*, pages 307–313.

Sugasawa, S. and Kubokawa, T. (2015). Box-Cox transformed linear mixed models for positive-valued and clustered data. *Manuscript*.

Trovato, G. and Caiazza, S. (2004). Extending Logistic Approach to Risk Modelling Through Semiparametric Mixing. Technical report, Tor Vergata University, CEIS.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Verbeke, G. and Molenberghs, G. (2013). The gradient function as an exploratory goodness-of-fit assessment of the random-effects distribution in mixed models. *Biostatistics*, 14(3):477–490.

Wang, L. (2004). *Parameter estimation for mixtures of generalized linear mixed-effects models*. PhD thesis, University of Georgia.

Wang, P., Tsai, G. f., and Qu, A. (2012). Conditional inference functions for mixed-effects models with unspecified random-effects distribution. *Journal of the American Statistical Association*, 107(498):725–736.

Xu, C., Baines, P. D., and Wang, J. L. (2014). Standard error estimation using the

EM algorithm for the joint modeling of survival and longitudinal data. *Biostatistics*, 15(4):731–744.

Appendix A

Appendix A

A.1 R codes for the simulation studies

The following R codes are used for the simulation studies in Chapters 2, 3, 4 and 5, respectively:

A.1.1 Box-Cox transformations for random effect models

R Note:

Simulation study 1:

```
library(boxcoxmix)


#Simulation using fixed lambda


beta1 <- 3
```

```
beta2 <- 0.5

beta  <- c(beta1,beta2)

n<-100 ## sample size

N <-1000  # number of simulation runs

save.coef1<-matrix(0,N,2)

save.coef2<-matrix(0,N,2)

save.coef3<-matrix(0,N,2)

save.coef4<-matrix(0,N,2)

save.se1<-matrix(0,N,2)

save.se2<-matrix(0,N,2)

save.se3<-matrix(0,N,2)

save.se4<-matrix(0,N,2)

mass.point <- c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(j in 1:N){

  x1 <- runif(n, min = -1, max = 1)

  x2 <- runif(n, min = -3, max = 3)

  x<-cbind(x1,x2)

  error <- rnorm(n, mean = 0, sd = 0.5)

  z    <- sample(mass.point,size=n,mass,replace=TRUE)

  # Generate the response eta_ij that is normally distributed

  eta1 <- x%*% beta +z+error ## this will be used in Section A.2
```

```
zeta1 <- yhat(eta1,0)

zeta2 <- yhat(eta1,0.5)

zeta3 <- yhat(eta1,1)

zeta4 <- yhat(eta1,2)

dat1 <- cbind(eta1,zeta1,zeta2,zeta3,zeta4,x1,x2)

dat1 <- as.data.frame(dat1)

colnames(dat1)<-c("eta1","zeta1","zeta2","zeta3","zeta4",

"x1","x2")

fit1 <- np.boxcoxmix(zeta1~x1+x2, data = dat1,lambda = 0,

                     K = 4, plot.opt=0, verbose = FALSE,

                      na.print=TRUE, start = "gq")

save.coef1[j,]<-fit1$beta

save.se1[j,]<-fit1$se

fit2 <-  np.boxcoxmix(zeta2~x1+x2, data = dat1,lambda = 0.5,

                       K = 4, plot.opt=0, verbose = FALSE,

                        na.print=TRUE, start = "gq")

save.coef2[j,]<-fit2$beta

save.se2[j,]<-fit2$se

fit3 <- np.boxcoxmix(zeta3~x1+x2, data = dat1, lambda = 1,

                     K = 4, plot.opt=0, verbose = FALSE,

                      na.print=TRUE, start = "gq")

save.coef3[j,]<-fit3$beta

save.se3[j,]<-fit3$se
```

```
  fit4 <- np.boxcoxmix(zeta4~x1+x2, data = dat1, lambda = 2,
                          K = 4, plot.opt=0, verbose = FALSE,
                           na.print=TRUE, start = "gq")
  save.coef4[j,]<-fit4$beta
  save.se4[j,]<-fit4$se
  sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,
   save.se1, save.se2, save.se3, save.se4,
   file = "~/Desktop/re/resmlfix.Rdata")
}
```

```
#Simulation using unknown lambda


beta1 <- 3
beta2 <- 0.5
beta  <- c(beta1,beta2)
n<-100 ## sample size
N <-1000  # number of simulation runs
save.lambda1<-rep(0,N)
save.lambda2<-rep(0,N)
save.lambda3<-rep(0,N)
save.lambda4<-rep(0,N)
save.coef1<-matrix(0,N,2)
save.coef2<-matrix(0,N,2)
```

```
save.coef3<-matrix(0,N,2)

save.coef4<-matrix(0,N,2)

save.se1<-matrix(0,N,2)

save.se2<-matrix(0,N,2)

save.se3<-matrix(0,N,2)

save.se4<-matrix(0,N,2)

mass.point <- c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(j in 1:N){

  x1 <- runif(n, min = -1, max = 1)

  x2 <- runif(n, min = -3, max = 3)

  x<-cbind(x1,x2)

  error <- rnorm(n, mean = 0, sd = 0.5)

  z    <- sample(mass.point,size=n,mass,replace=TRUE)

  # Generate the response eta_ij that is normally distributed

  eta1 <- x%*% beta +z+error ## this will be used in Section A.2

  zeta1 <- yhat(eta1,0)

  zeta2 <- yhat(eta1,0.5)

  zeta3 <- yhat(eta1,1)

  zeta4 <- yhat(eta1,2)

  dat1 <- cbind(eta1,zeta1,zeta2,zeta3,zeta4,x1,x2)

  dat1 <- as.data.frame(dat1)
```

```
colnames(dat1)<-c("eta1","zeta1","zeta2","zeta3","zeta4",
"x1","x2")
fit1 <- optim.boxcox(zeta1~x1+x2, data = dat1,
find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,
 verbose = FALSE, na.print=TRUE, start = "gq")
save.coef1[j,]<-fit1$beta
save.se1[j,]<-fit1$se
save.lambda1[j]<-fit1$Maximum
fit2 <- optim.boxcox(zeta2~x1+x2, data = dat1,
find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,
 verbose = FALSE, na.print=TRUE, start = "gq")
save.coef2[j,]<-fit2$beta
save.se2[j,]<-fit2$se
save.lambda2[j]<-fit2$Maximum
fit3 <- optim.boxcox(zeta3~x1+x2, data = dat1,
find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,
 verbose = FALSE, na.print=TRUE, start = "gq")
save.coef3[j,]<-fit3$beta
save.se3[j,]<-fit3$se
save.lambda3[j]<-fit3$Maximum
fit4 <- optim.boxcox(zeta4~x1+x2, data = dat1,
find.in.range = c(-.2, 3), K = 4, plot.opt=0, s=32,
verbose = FALSE, na.print=TRUE, start = "gq")
```

```
  save.coef4[j,]<-fit4$beta

  save.se4[j,]<-fit4$se

  save.lambda4[j]<-fit4$Maximum

  sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,

   save.se1, save.se2, save.se3, save.se4,save.lambda1,

    save.lambda2, save.lambda3, save.lambda4,

    file = "~/Desktop/re/resmlopt.Rdata")

}
```

R Note:

Simulation study 2:

```
#Simulation using fixed lambda



beta1 <- 5

beta2 <- 3

beta  <- c(beta1,beta2)

n<-100 ## sample size

N <-1000  # number of simulation runs

save.coef1<-matrix(0,N,2)

save.coef2<-matrix(0,N,2)

save.coef3<-matrix(0,N,2)

save.coef4<-matrix(0,N,2)
```

```
save.se1<-matrix(0,N,2)

save.se2<-matrix(0,N,2)

save.se3<-matrix(0,N,2)

save.se4<-matrix(0,N,2)

mass.point <-c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(j in 1:N){

  x1 <- runif(n, min = -1, max = 1)

  x2 <- runif(n, min = 0, max = 4)

  x<-cbind(x1,x2)

  error <- rnorm(n, mean = 0, sd = 0.5)

  z    <- sample(mass.point,size=n,mass,replace=TRUE)

  # Generate the response eta_ij that is normally distributed

  eta2 <- x%*% beta +z+error ## this will be used in Section A.2

  zeta1 <- yhat(eta2,0)

  zeta2 <- yhat(eta2,0.5)

  zeta3 <- yhat(eta2,1)

  zeta4 <- yhat(eta2,2)

  dat2 <- cbind(eta2,zeta1,zeta2,zeta3,zeta4,x1,x2)

  dat2 <- as.data.frame(dat2)

  colnames(dat2)<-c("eta2","zeta1","zeta2","zeta3","zeta4",

  "x1","x2")
```

```
    fit1 <- np.boxcoxmix(zeta1~x1+x2, data = dat2,lambda = 0,
                         K = 4, plot.opt=0, verbose = FALSE,
                          na.print=TRUE, start = "gq")
  save.coef1[j,]<-fit1$beta
  save.se1[j,]<-fit1$se
  fit2 <-  np.boxcoxmix(zeta2~x1+x2, data = dat2,lambda = 0.5,
                           K = 4, plot.opt=0, verbose = FALSE,
                            na.print=TRUE, start = "gq")
  save.coef2[j,]<-fit2$beta
  save.se2[j,]<-fit2$se
  fit3 <- np.boxcoxmix(zeta3~x1+x2, data = dat2, lambda = 1,
                         K = 4, plot.opt=0, verbose = FALSE,
                          na.print=TRUE, start = "gq")
  save.coef3[j,]<-fit3$beta
  save.se3[j,]<-fit3$se
  fit4 <- np.boxcoxmix(zeta4~x1+x2, data = dat2, lambda = 2,
                         K = 4, plot.opt=0, verbose = FALSE,
                          na.print=TRUE, start = "gq")
  save.coef4[j,]<-fit4$beta
  save.se4[j,]<-fit4$se
  sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,
   save.se1, save.se2, save.se3, save.se4,
    file = "~/Desktop/re2/resmlfix2.Rdata")
}
```

```
#Simulation using unknown lambda



beta1 <- 5

beta2 <- 3

beta  <- c(beta1,beta2)

n<-100 ## sample size

N <-1000  # number of simulation runs

save.lambda1<-rep(0,N)

save.lambda2<-rep(0,N)

save.lambda3<-rep(0,N)

save.lambda4<-rep(0,N)

save.coef1<-matrix(0,N,2)

save.coef2<-matrix(0,N,2)

save.coef3<-matrix(0,N,2)

save.coef4<-matrix(0,N,2)

save.se1<-matrix(0,N,2)

save.se2<-matrix(0,N,2)

save.se3<-matrix(0,N,2)

save.se4<-matrix(0,N,2)

mass.point <-c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(j in 1:N){

  x1 <- runif(n, min = -1, max = 1)
```

```
x2 <- runif(n, min = 0, max = 4)

x<-cbind(x1,x2)

error <- rnorm(n, mean = 0, sd = 0.5)

z    <- sample(mass.point,size=n,mass,replace=TRUE)

# Generate the response eta_ij that is normally distributed

eta2 <- x%*% beta +z+error ## this will be used in Section A.2

zeta1 <- yhat(eta2,0)

zeta2 <- yhat(eta2,0.5)

zeta3 <- yhat(eta2,1)

zeta4 <- yhat(eta2,2)

dat2 <- cbind(eta2,zeta1,zeta2,zeta3,zeta4,x1,x2)

dat2 <- as.data.frame(dat2)

colnames(dat2)<-c("eta2","zeta1","zeta2","zeta3","zeta4",

"x1","x2")

fit1 <- optim.boxcox(zeta1~x1+x2, data = dat2,

find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

verbose = FALSE, na.print=TRUE, start = "gq")

save.coef1[j,]<-fit1$beta

save.se1[j,]<-fit1$se

save.lambda1[j]<-fit1$Maximum

fit2 <- optim.boxcox(zeta2~x1+x2, data = dat2,

find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

verbose = FALSE, na.print=TRUE, start = "gq")
```

```
  save.coef2[j,]<-fit2$beta

  save.se2[j,]<-fit2$se

  save.lambda2[j]<-fit2$Maximum

  fit3 <- optim.boxcox(zeta3~x1+x2, data = dat2,

  find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

  verbose = FALSE, na.print=TRUE, start = "gq")

  save.coef3[j,]<-fit3$beta

  save.se3[j,]<-fit3$se

  save.lambda3[j]<-fit3$Maximum

  fit4 <- optim.boxcox(zeta4~x1+x2, data = dat2,

  find.in.range = c(-.2, 3), K = 4, plot.opt=0, s=32,

  verbose = FALSE, na.print=TRUE, start = "gq")

  save.coef4[j,]<-fit4$beta

  save.se4[j,]<-fit4$se

  save.lambda4[j]<-fit4$Maximum

  sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,

   save.se1, save.se2, save.se3, save.se4,save.lambda1,

    save.lambda2, save.lambda3, save.lambda4,

     file = "~/Desktop/re2/resmlopt2.Rdata")

}
```

A.1.2 Box-Cox transformations for two–level models

R Note:

```
#Simulation using fixed lambda



beta <- 3

r   <- 20          # No. of upper-level groups

n_i <- rep(5, r)   # No. of lower-level within i-th group

n <- sum(n_i) #  sample size

N <-1000  # number of simulation runs

save.coef1<-rep(0,N)

save.coef2<-rep(0,N)

save.coef3<-rep(0,N)

save.coef4<-rep(0,N)

save.se1<-rep(0,N)

save.se2<-rep(0,N)

save.se3<-rep(0,N)

save.se4<-rep(0,N)

mass.point <- c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(s in 1:N){

    x_ij <- runif(n, min = -4, max = 4)

  Xbeta <-beta * x_ij
```

```
# Generate e_ij from a normal.

e_ij<-rnorm(n, mean = 0, sd = 0.5)

#random effect

z <- sample(mass.point,size=r,mass,replace=TRUE)

#the same cluster has the same random effect

z_i<-rep(z, n_i)

# Generate the response eta_ij that is normally distributed

eta_ij <- Xbeta +z_i+e_ij

zeta1 <- yhat(eta_ij ,0)

zeta2 <- yhat(eta_ij ,0.5)

zeta3 <- yhat(eta_ij ,1)

zeta4 <- yhat(eta_ij ,2)

gr <- gl(20,5) # groups

dat <- cbind(eta_ij ,zeta1,zeta2,zeta3,zeta4,x_ij,gr)

dat <- as.data.frame(dat)

colnames(dat)<-c("eta","zeta1","zeta2","zeta3","zeta4","X","gr")

fit1 <- np.boxcoxmix(zeta1~X, groups = dat$gr , data = dat,

 plot.opt=0, lambda = 0, K = 4,  verbose = FALSE,

  na.print=TRUE, start = "gq")

save.coef1[s]<-fit1$beta

save.se1[s]<-fit1$se

fit2 <- np.boxcoxmix(zeta2~X, groups = dat$gr , data = dat,

 plot.opt=0, lambda = 0.5,  K = 4,  verbose = FALSE,
```

```
    na.print=TRUE, start = "gq")
  save.coef2[s]<-fit2$beta
  save.se2[s]<-fit2$se
  fit3 <- np.boxcoxmix(zeta3~X, groups = dat$gr , data = dat,
   plot.opt=0, lambda = 1, K = 4,  verbose = FALSE,
    na.print=TRUE, start = "gq")
  save.coef3[s]<-fit3$beta
  save.se3[s]<-fit3$se
  fit4 <- np.boxcoxmix(zeta4~X, groups = dat$gr , data = dat,
   plot.opt=0, lambda = 2, K = 4,  verbose = FALSE,
    na.print=TRUE, start = "gq")
  save.coef4[s]<-fit4$beta
  save.se4[s]<-fit4$se
sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,
 save.se1, save.se2, save.se3, save.se4,
  file = "~/Desktop/vc/vcsmlfix.Rdata")
}
```

```
#Simulation using unknown lambda


beta <- 3
r   <- 20          # No. of upper-level groups
```

```
n_i <- rep(5, r)   # No. of lower-level within i-th group

n <- sum(n_i) #  sample size

N <-1000  # number of simulation runs

save.lambda1<-rep(0,N)

save.lambda2<-rep(0,N)

save.lambda3<-rep(0,N)

save.lambda4<-rep(0,N)

save.coef1<-rep(0,N)

save.coef2<-rep(0,N)

save.coef3<-rep(0,N)

save.coef4<-rep(0,N)

save.se1<-rep(0,N)

save.se2<-rep(0,N)

save.se3<-rep(0,N)

save.se4<-rep(0,N)

mass.point <- c(15,20,30,35)

K <- 4

mass <- rep(1/K, K)

for(s in 1:N){

    x_ij <- runif(n, min = -4, max = 4)

  Xbeta <-beta * x_ij

  # Generate e_ij from a normal.

  e_ij<-rnorm(n, mean = 0, sd = 0.5)

  #random effect
```

```
z <- sample(mass.point,size=r,mass,replace=TRUE)

#the same cluster has the same random effect

z_i<-rep(z, n_i)

# Generate the response eta_ij that is normally distributed

eta_ij <- Xbeta +z_i+e_ij

zeta1 <- yhat(eta_ij ,0)

zeta2 <- yhat(eta_ij ,0.5)

zeta3 <- yhat(eta_ij ,1)

zeta4 <- yhat(eta_ij ,2)

gr <- gl(20,5) # groups

dat <- cbind(eta_ij ,zeta1,zeta2,zeta3,zeta4,x_ij,gr)

dat <- as.data.frame(dat)

colnames(dat)<-c("eta","zeta1","zeta2","zeta3","zeta4","X","gr")

fit1 <- optim.boxcox(zeta1~X, groups = dat$gr ,data = dat,

find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

verbose = FALSE, na.print=TRUE, start = "gq")

save.coef1[s]<-fit1$beta

save.se1[s]<-fit1$se

save.lambda1[s]<-fit1$Maximum

fit2 <- optim.boxcox(zeta2~X, groups = dat$gr ,data = dat,

find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

verbose = FALSE, na.print=TRUE, start = "gq")

save.coef2[s]<-fit2$beta

save.se2[s]<-fit2$se
```

```
  save.lambda2[s]<-fit2$Maximum

  fit3 <- optim.boxcox(zeta3~X, groups = dat$gr ,data = dat,

  find.in.range = c(-.2, 2), K = 4, plot.opt=0, s=22,

  verbose = FALSE, na.print=TRUE, start = "gq")

  save.coef3[s]<-fit3$beta

  save.se3[s]<-fit3$se

  save.lambda3[s]<-fit3$Maximum

  fit4 <- optim.boxcox(zeta4~X, groups = dat$gr ,data = dat,

  find.in.range = c(-.2, 3), K = 4, plot.opt=0, s=32,

  verbose = FALSE, na.print=TRUE, start = "gq")

  save.coef4[s]<-fit4$beta

  save.se4[s]<-fit4$se

  save.lambda4[s]<-fit4$Maximum

sml2opt<-save(save.coef1, save.coef2, save.coef3, save.coef4,

 save.se1, save.se2, save.se3, save.se4,save.lambda1,

  save.lambda2, save.lambda3, save.lambda4,

   file = "~/Desktop/vc/vcsmlopt.Rdata")


}
```

A.1.3 Transformations for fixed–effect binary regression models

R Note:

```
#Simulation using fixed lambda



beta0 <- 2

beta1 <- 1

beta  <- c(beta0,beta1)

n <- 100   # sample size

N <-1000   # number of simulation runs

m<-40     # number of trails

save.coefg<-matrix(0,N,2)

save.coef0<-matrix(0,N,2)

save.coef1<-matrix(0,N,2)

save.coef2<-matrix(0,N,2)

save.coef3<-matrix(0,N,2)

save.coef4<-matrix(0,N,2)

for(j in 1:N){

  x1 <- runif(n, min = -1, max = 1)

  etai<- cbind(1,x1)%*% beta

  pi0<- exp(etai)/(1+exp(etai))  #lambda=0

  pi1<- ((1-0.2*etai)^(-1/-0.2)+1)^(-1) #lambda=-0.2

  pi2<- ((1+0.2*etai)^(-1/0.2)+1)^(-1) #lambda=0.2
```

```
pi3<- ((1+0.5*etai)^(-1/0.5)+1)^(-1) #lambda=0.5

pi4<- ((1+1*etai)^(-1/1)+1)^(-1) #lambda=1

y0<-rbinom(n,m, pi0)

y1<-rbinom(n,m, pi1)

y2<-rbinom(n,m, pi2)

y3<-rbinom(n,m, pi3)

y4<-rbinom(n,m, pi4)

dat <- cbind(y0,y1,y1,y2,y3,y4,x1)

dat <- as.data.frame(dat)

fitg <- alldist(y0/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=logit),k=1)

fit0 <- alldist( y0/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=boxcoxpower(0)),k=1)

fit1 <- alldist(y1/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=boxcoxpower(-0.2)),k=1)

fit2 <- alldist( y2/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=boxcoxpower(0.2)),k=1)

fit3 <- alldist( y3/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=boxcoxpower(0.5)),k=1)

fit4 <- alldist(y4/40~x1, weights = rep(40, 100),data = dat,

 family=binomial(link=boxcoxpower(1)),k=1)

save.coefg[j,]<- fitg$coef

save.coef0[j,]<-fit0$coef
```

```
  save.coef1[j,]<-fit1$coef

  save.coef2[j,]<-fit2$coef

  save.coef3[j,]<-fit3$coef

  save.coef4[j,]<-fit4$coef

  smlfixb<-save(save.coefg,save.coef0,save.coef1,

  save.coef2, save.coef3, save.coef4,

    file = "~/Desktop/fixedbinary/smlfix.Rdata")

}
```

```
#Simulation using unknown lambda



beta0 <- 2

beta1 <- 1

beta  <- c(beta0,beta1)

n <- 100   # sample size

N <-1000   # number of simulation runs

m<-40    # number of trails

save.coeff0<-matrix(0,N,2)

save.coeff1<-matrix(0,N,2)

save.coeff2<-matrix(0,N,2)

save.coeff3<-matrix(0,N,2)

save.coeff4<-matrix(0,N,2)
```

```
save.llambda0<-rep(0,N)

save.llambda1<-rep(0,N)

save.llambda2<-rep(0,N)

save.llambda3<-rep(0,N)

save.llambda4<-rep(0,N)

for(j in 1:N){

  print(j)

  x1 <- runif(n, min = -1, max = 1)

  etai<- cbind(1,x1)%*% beta

  pi0<- exp(etai)/(1+exp(etai)) #lambda=0

  pi1<- ((1-0.2*etai)^(-1/-0.2)+1)^(-1)  #lambda=-0.2

  pi2<- ((1+0.2*etai)^(-1/0.2)+1)^(-1) #lambda=0.2

  pi3<- ((1+0.5*etai)^(-1/0.5)+1)^(-1) #lambda=0.5

  pi4<- ((1+1*etai)^(-1/1)+1)^(-1) #lambda=1

  y0<-rbinom(n,m, pi0)

  y1<-rbinom(n,m, pi1)

  y2<-rbinom(n,m, pi2)

  y3<-rbinom(n,m, pi3)

  y4<-rbinom(n,m, pi4)

  dat <- cbind(y0,y1,y2,y3,y4,x1)

  dat <- as.data.frame(dat)

  #colnames(dat)<-c("y0","y1","y2","y3","y4","x1")

  fit0 <- boxcoxtype(y0/40~x1,data=dat, trials = 40,
```

```
    find.in.range = c(-0.3, 1.2), s = 16,k=1)

   save.coeff0[j,]<-fit0$coef

   save.llambda0[j]<-fit0$Maximum

   fit1 <-  boxcoxtype(y1/40~x1,data=dat, trials = 40,

    find.in.range = c(-0.4, 1), s = 16,k=1)

   save.coeff1[j,]<- fit1$coef

   save.llambda1[j]<-fit1$Maximum

   fit2 <-  boxcoxtype(y2/40~x1,data=dat, trials = 40,

    find.in.range = c(-0.2, 1.3), s = 16,k=1)

   save.coeff2[j,]<- fit2$coef

   save.llambda2[j]<-fit2$Maximum

   fit3 <- boxcoxtype(y3/40~x1,data=dat, trials = 40,

    find.in.range = c(-0.2, 1.5), s = 18,k=1)

   save.coeff3[j,]<- fit3$coef

   save.llambda3[j]<-fit3$Maximum

   fit4 <- boxcoxtype(y4/40~x1,data=dat, trials = 40,

   find.in.range = c(-0.2, 1.6), s = 19,k=1)

   save.coeff4[j,]<- fit4$coef

   save.llambda4[j]<-fit4$Maximum

   sml2opt1000b<-save(save.coeff0,save.coeff1,save.coeff2,

   save.coeff3,save.coeff4, save.llambda0,save.llambda1,

   save.llambda2,save.llambda3, save.llambda4,

   file = "~/Desktop/fixedbinary/smlopt.Rdata")

 }
```

A.1.4 Transformations for mixed–effects binary regression models

R Note:

```
#Simulation using fixed lambda



beta1 <-3

beta2 <- 0.5

beta  <- c(beta1,beta2)##

n<- 100 ## sample size

N <-1000   # number of simulation runs

m<-40    # number of trails

save.coefg<-save.coef0<-matrix(0,N,2)

save.coef1<-save.coef2<-save.coef3<-matrix(0,N,2)

mass.point <- c(35,60,5)

K <- 3 #number of clusters

mass <- rep(1/K, K)

for(j in 1:N){

  print(j)

  x1 <- runif(n, min = -1, max =1)

  x2 <- runif(n, min = -1, max =1)

  x<-cbind(x1,x2)

  z    <- sample(mass.point,size=n,mass,replace=TRUE)

  etai<- x%*% beta + z
```

```
pi0<- exp(etai)/(1+exp(etai)) #lambda=0

pi1<- ((1+0.2*etai)^(-1/0.2)+1)^(-1) #lambda=0.2

pi2<- ((1+0.5*etai)^(-1/0.5)+1)^(-1) #lambda=0.5

pi3<- ((1+1*etai)^(-1/1)+1)^(-1) #lambda=1

y0<-rbinom(n,m, pi0)

y1<-rbinom(n,m, pi1)

y2<-rbinom(n,m, pi2)

y3<-rbinom(n,m, pi3)

dat <- cbind(y0,y1,y2,y3,x1,x2,z)

dat <- as.data.frame(dat)

fitg <- alldist( y0/40~x1+x2,k=3, weights = rep(40, 100),

data = dat, verbose=FALSE, family=binomial(link=logit),

 plot.opt = 0)

save.coefg[j,]<- fitg$coefficients[1:abs(length

(fitg$coefficients)-length(fitg$mass.points))]

fit0 <- alldist( y0/40~x1+x2, k=3,weights = rep(40, 100),

 data = dat, verbose=FALSE, family=binomial(link=boxcoxpower(0)),

plot.opt = 0)

save.coef0[j,]<-fit0$coefficients[1:abs(length

(fit0$coefficients)-length(fit0$mass.points))]

fit1 <-  alldist( y1/40~x1+x2,k=3, weights = rep(40, 100),

 data = dat, verbose=FALSE, family=binomial(link=

 boxcoxpower(0.2)), plot.opt = 0)
```

```
  save.coef1[j,]<- fit1$coefficients[1:abs(length
  (fit1$coefficients)-length(fit1$mass.points))]
  fit2 <-  alldist( y2/40~x1+x2, k=3,weights = rep(40, 100),
   data = dat, verbose=FALSE, family=binomial(link=
   boxcoxpower(0.5)), plot.opt = 0)
  save.coef2[j,]<- fit2$coefficients[1:abs(length
  (fit2$coefficients)-length(fit2$mass.points))]
  fit3 <- alldist( y3/40~x1+x2,k=3, weights = rep(40, 100),
   data = dat, verbose=FALSE, family=binomial(link=boxcoxpower(1)),
   plot.opt = 0)
  save.coef3[j,]<- fit3$coefficients[1:abs(length
  (fit3$coefficients)-length(fit3$mass.points))]
  smlmixfixb<-save(save.coefg,save.coef0,save.coef1,save.coef2,
  save.coef3, file = "~/Desktop/mixedbinary/smlmixfix.Rdata")
}
```

```
#Simulation using unknown lambda


beta1 <-3 #3
beta2 <- 0.5
beta  <- c(beta1,beta2)##
n<- 100 ## sample size
N <-1000   # number of simulation runs
```

```
m<-40    # number of trails

save.coef0<-save.coef1<-save.coef2<-save.coef3<-matrix(0,N,2)

save.llambda0<-save.llambda1<-rep(0,N)

save.llambda2<-save.llambda3<-rep(0,N)

mass.point <- c(35,60,5)

K <- 3 #number of clusters

mass <- rep(1/K, K)

for(j in 1:N){

  x1 <- runif(n, min = -1, max =1)

  x2 <- runif(n, min = -1, max =1)

  x<-cbind(x1,x2)

  z    <- sample(mass.point,size=n,mass,replace=TRUE)

  etai<- x%*% beta + z

  pi0<- exp(etai)/(1+exp(etai)) #lambda=0

  pi1<- ((1+0.2*etai)^(-1/0.2)+1)^(-1) #lambda=0.2

  pi2<- ((1+0.5*etai)^(-1/0.5)+1)^(-1) #lambda=0.5

  pi3<- ((1+1*etai)^(-1/1)+1)^(-1) #lambda=1

  y0<-rbinom(n,m, pi0)

  y1<-rbinom(n,m, pi1)

  y2<-rbinom(n,m, pi2)

  y3<-rbinom(n,m, pi3)

  dat <- cbind(y0,y1,y2,y3,x1,x2,z)

  dat <- as.data.frame(dat)
```

```
fit0 <- boxcoxtype(y0/40~x1+x2,k=3,data=dat,

trials = 40, find.in.range = c(-0.1,1), s = 12)

save.coef0[j,]<-fit0$coef

save.llambda0[j]<-fit0$Maximum

fit1 <-  boxcoxtype(y1/40~x1+x2,k=3,data=dat,

 trials = 40, find.in.range = c(-0.1,1), s = 12)

save.coef1[j,]<- fit1$coef

save.llambda1[j]<-fit1$Maximum

fit2 <-  boxcoxtype(y2/40~x1+x2,k=3,data=dat,

trials = 40,  find.in.range = c(-0.1,1.5), s = 16)

save.coef2[j,]<- fit2$coef

save.llambda2[j]<-fit2$Maximum

fit3 <-  boxcoxtype(y3/40~x1+x2,k=3,data=dat,

 trials = 40,  find.in.range = c(-0.1,1.5), s = 16)

save.coef3[j,]<- fit3$coef

save.llambda3[j]<-fit3$Maximum

smlmixoptb<-save(save.coef0,save.coef1,save.coef2,save.coef3,

 save.llambda0, save.llambda1, save.llambda2, save.llambda3,

  file = "~/Desktop/mixedbinary/smlmixopt.Rdata")

}
```

A.2 A comparison of the simulation studies of the random effect models

In this section, the results from the fit of the random effect model to the simulated data in their original forms (i.e. without transformation) for the two studies presented in Section 2.7 are shown for comparison. In the simulation step, we need to ensure that the simulated data η_i is normally distributed before applying the inverse of the transformation. An assessment of the normality of data can be carried out using a graphical method. The normal probability plots (QQ–plot) and histograms are used to represent the distribution of the data. In the histogram, the data follows a normal distribution if its points represented as a bell-shaped curve. In the following graphs, we attempt to illustrate the effects of the design of the data on its distribution's shape using the function `np.boxcoxmix()` in `R`, setting $\lambda = 1$ (i.e. no transformation). From the outputs, we plot a histogram and a QQ–plot for the residuals of fitted model η_i, which are expressed as $\hat{\varepsilon}_i = \eta_i - \hat{\eta}_i = \eta_i - x_i^T\hat{\beta} - \hat{z}_i$, where $\hat{z}_i = \sum_{k=1}^{K} w_{ik}\hat{z}_k$.

For the first simulation study, we generated the random sample from normal distribution for sample size 100 as

$$\eta_{i1} = 3\,x_{1,i} + 0.5\,x_{2,i} + z_i + \varepsilon_i \tag{A.2.1}$$

$$X_1 \sim U(-1, 1),\ X_2 \sim U(-3, 3)$$

$$\varepsilon \sim N(0, 0.5^2)$$

$$\lambda_1 = 0,\ \ \lambda_2 = 0.5,\ \ \lambda_3 = 1,\ \ \lambda_4 = 2$$

$$z_i \sim \text{Multinomial}\{1, (z_1, \ldots, z_4)|\pi_1, \ldots, \pi_4\}$$

$$z_k = (15, 20, 30, 35) \text{ with masses } \pi_k = 1/4,\ k = 1, ..., 4.$$

and for the second simulation study, we generated the random sample from normal distribution for sample size 100 as

$$\eta_{i2} = 5x_{1,i} + 3x_{2,i} + z_i + \varepsilon_i \tag{A.2.2}$$

$$X_1 \sim U(-1, 1),\ X_2 \sim U(0, 4)$$

$$\varepsilon \sim N(0, 0.5^2)$$

$$\lambda_1 = 0,\ \ \lambda_2 = 0.5,\ \ \lambda_3 = 1,\ \ \lambda_4 = 2$$

$$z_i \sim \text{Multinomial}\{1, (z_1, \ldots, z_4)|\pi_1, \ldots, \pi_4\}$$

$$z_k = (15, 20, 30, 35) \text{ with masses } \pi_k = 1/4,\ k = 1, ..., 4.$$

See Subsections A.1.1 and A.1.1. We now fit these models η_{i1} and η_{i2} given in (A.2.1) and (A.2.2) , respectively, using the **boxcoxmix** function `np.boxcoxmix()` with $\lambda = 1$ (*i.e.* no transformation).

R Note:

We obtain the residuals of the first study as follow,

```
test1 <- np.boxcoxmix(eta1 ~ x1+x2, data = dat1, K=4,lambda=1)

Res1<-test1$residuals
```

for the residuals of the second study we use,

```
test2 <- np.boxcoxmix(eta2 ~ x1+x2, data = dat2, K=4,lambda=1)

Res2<-test2$residuals
```

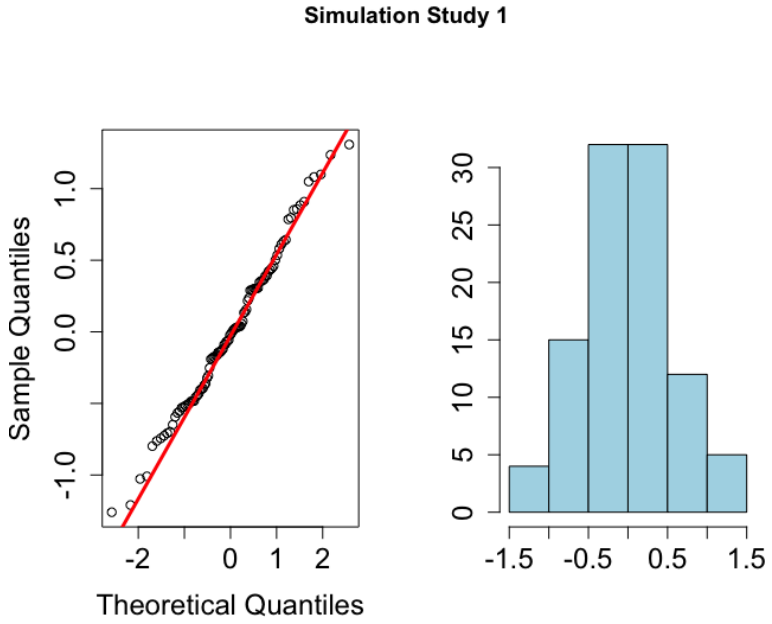


Figure A.2.1: Simulation Study 1: an assessment of the normality of the residuals for simulated data of the first study using QQ-plot and Histogram

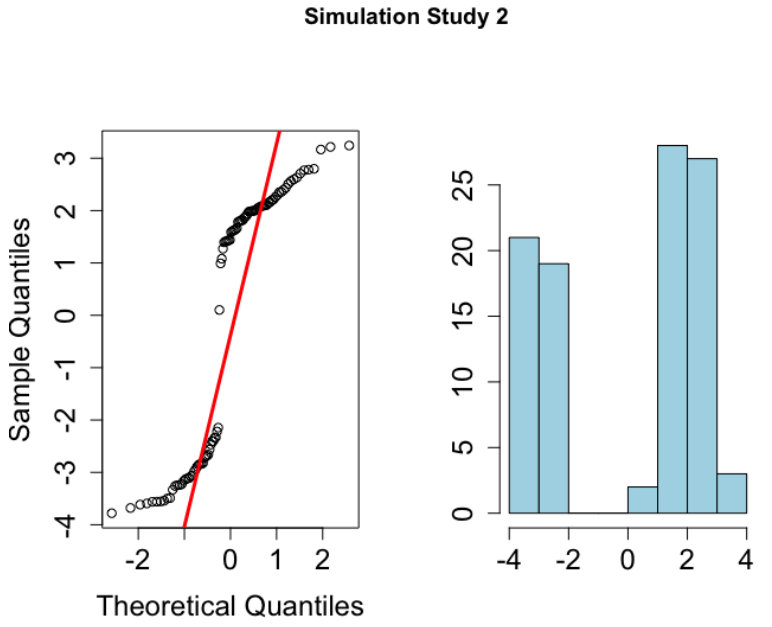


Figure A.2.2: Simulation Study 2: an assessment of the normality of the residuals for simulated data of the second study using QQ-plot and Histogram

Figure A.2.1 shows the results of the first study while the results of the second study are shown in Figure A.2.2. The QQ–plots for the residuals of simulated

data sets are shown on the left-hand side while the histograms are on the right-hand side of the Figures. From the two QQ–plots it appears that the model fit of the first study produces a normally distributed residuals while the model fit of the second study shows curvature at some points along the curve. The histograms confirm this since the histogram of the first study has a perfect bell-shaped while the histogram of the second study looks quite different from a bell. Note that this compression is before applying any transformation (neither backwards nor forwards).

A.3 Simulations using fixed λ

Transforming a data set backwards followed by a forwards transformation using the same value of λ means no transformation takes place. To prove that recall the equation for the 'forwards' Box-Cox transformation of the response y_i,

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & (\lambda \neq 0), \\ \log y_i & (\lambda = 0) \end{cases} \tag{A.3.1}$$

and that for $y_i > 0$, $i = 1, ..., n$. From that the 'backward' Box–Cox–transformation is

$$\hat{y}_i = \begin{cases} \left(1 + \lambda\eta_i\right)^{1/\lambda} & (\lambda \neq 0), \\ e^{\eta_i} & (\lambda = 0) \end{cases} \tag{A.3.2}$$

where $\eta_i = x_i^T\beta + z_i$. Now using the λ's values in Section 2.7, the 'backward' transformations for each value of λ are

$$\hat{y}_i = \begin{cases} e^{\eta_i} & (\lambda = 0), \\ \Big(1 + \dfrac{\eta_i}{2}\Big)^2 & (\lambda = 0.5), \\ \Big(1 + \eta_i\Big) & (\lambda = 1), \\ \Big(1 + 2\eta_i\Big)^{1/2} & (\lambda = 2) \end{cases} \tag{A.3.3}$$

Applying the Box–Cox transformation forwards to (A.3.3) using the same value of λ yields

$$y_i^{(\lambda)} = \begin{cases} \log e^{\eta_i} = \eta_i & (\lambda = 0), \\ \dfrac{\Big(\Big(1 + (\eta_i/2)\Big)^2\Big)^{1/2} - 1}{1/2} = 2\Big(\Big(1 + (\eta_i/2)\Big) - 1\Big) = & \\ = 2 + \eta_i - 2 = \eta_i & (\lambda = 0.5), \\ \Big(1 + \eta_i\Big) - 1 = \eta_i & (\lambda = 1), \\ \dfrac{\Big(\Big(1 + 2\eta_i\Big)^{1/2}\Big)^2 - 1}{2} = \dfrac{\Big(1 + 2\eta_i\Big) - 1}{2} = & \\ = \dfrac{2\eta_i}{2} = \eta_i & (\lambda = 2) \end{cases} \tag{A.3.4}$$

This is the reason for having identical boxplots for different values of λ in Figures (2.7.2), (2.7.5) and (3.5.1). Note that this is not the case for the binary regression models in Figures (4.5.1) and (5.5.1) because we generated the data after applying the 'backward' Box–Cox transformation to the success probabilities for each value of λ whereas in the linear models we started with generating a data set then we applied the 'backward' Box–Cox transformation to this data for each value of λ (see Figures A.3.1 and A.3.2).

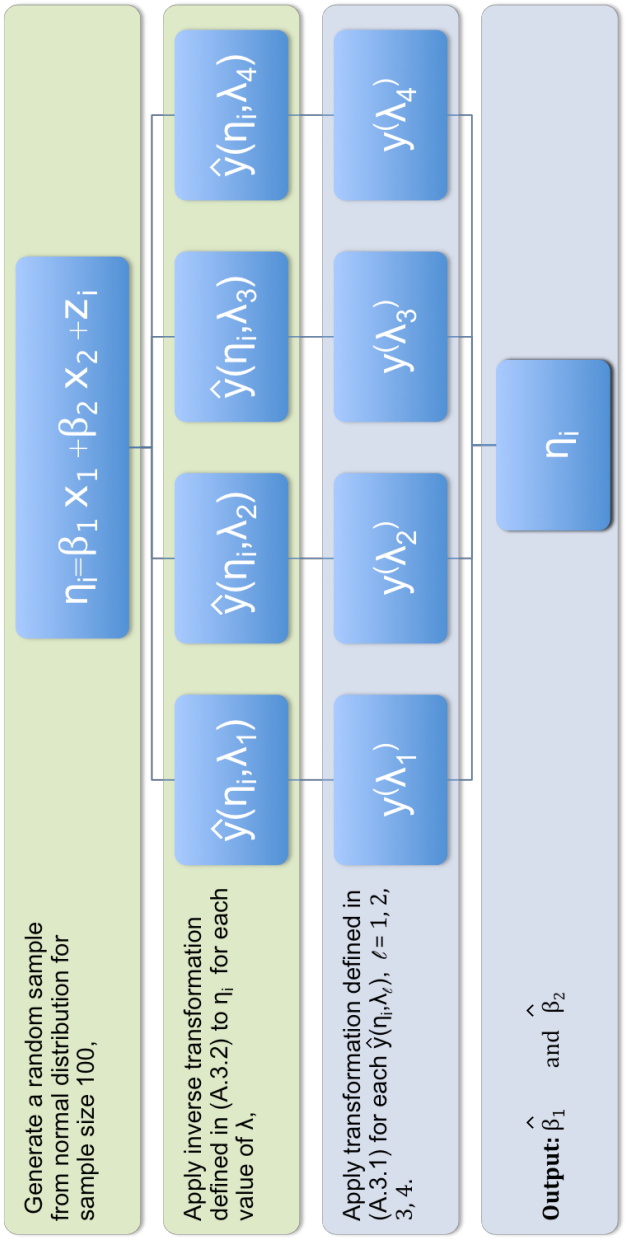


Figure A.3.1: Algorithm for simulation studies for the linear models with fixed λ values

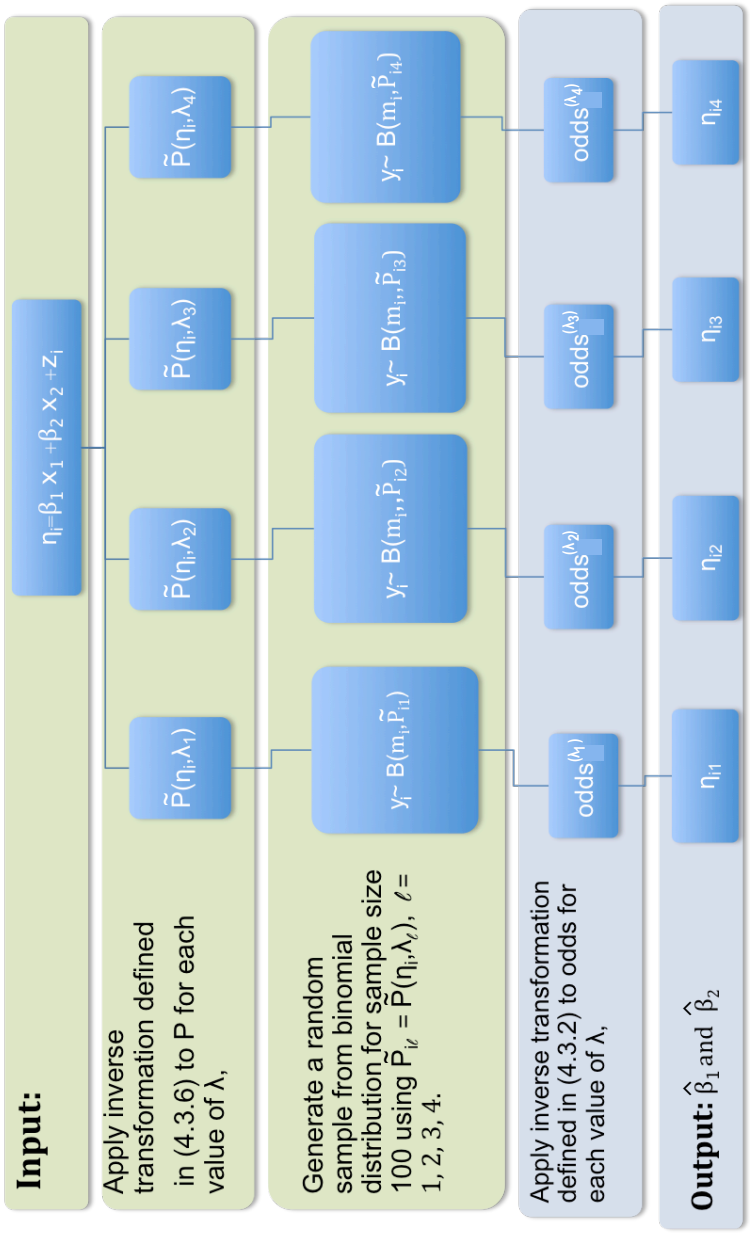


Figure A.3.2: Algorithm for simulation studies for the binary models with fixed λ values

A.4 Residual Plots

Figure A.4.1 shows some residuals plots for `WWWusage` data before and after applying the response transformation for $K \in [1, 4]$ (see Example 2.11.2).

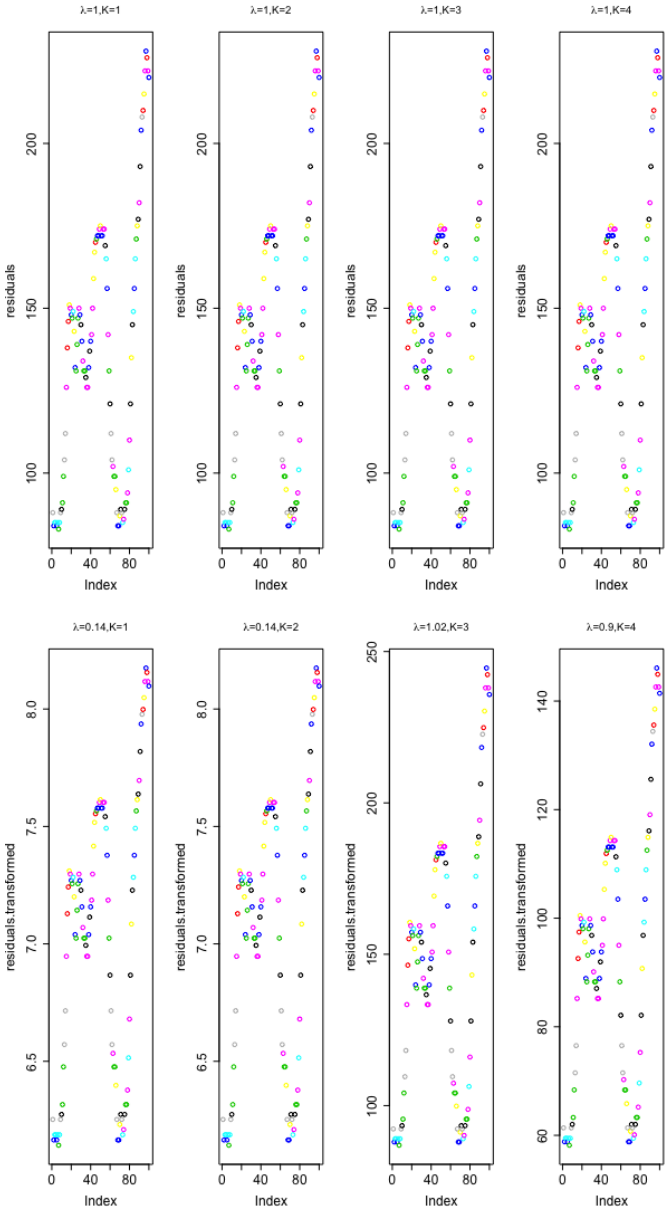


Figure A.4.1: The residuals plots for `WWWusage` data before and after applying the response transformation for $K \in [1, 4]$

Package 'boxcoxmix'

June 5, 2018

Type Package

Title Box-Cox-Type Transformations for Linear and Logistic Models with Random Effects

Version 0.20

Date 2018-6-5

Author Amani Almohaimeed and Jochen Einbeck

Maintainer Amani Almohaimeed <amani.almohaimeed@gmail.com>

Depends R (>= 3.3.0)

Imports statmod(>= 1.4.27), qicharts(>= 0.5.4), npmlreg(>= 0.46-1)

Suggests nlme, mdscore, flexmix, utils

LazyLoad yes

Description Box-Cox-type transformations for linear and logistic models with random effects using non-parametric profile maximum likelihood estimation. The main functions are optim.boxcox() and boxcoxtype().

License GPL (>=3)

RoxygenNote 6.0.1

VignetteBuilder utils

R topics documented:

boxcoxmix-package *Box-Cox-Type Transformations for Linear and Logistic Models with Random Effects*

Description

Box-Cox-type transformations for linear and logistic models with random effects using non-parametric profile maximum likelihood estimation. The main functions are `optim.boxcox()` and `boxcoxtype()`.

Details

Package:	boxcoxmix
Type:	Package
Version:	0.20
Date:	2018-6-5
License:	GPL (>=3)

Author(s)

Amani Almohaimeed and Jochen Einbeck

References

Box G. and Cox D. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), pages 211-252.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. (2009). Statistical modelling in R. Oxford University Press Oxford.

Jochen Einbeck, Ross Darnell and John Hinde (2014). npmlreg: Nonparametric maximum likelihood estimation for random effect models. R package version 0.46-1.

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Maintainer: Amani Almohaimeed <amani.almohaimeed@gmail.com>

boxcoxtype *Box-Cox-type link function for logistic mixed-effects Models*

Description

The `boxcoxtype()` performs a grid search over the parameter `Lambda` for logistic mixed-effects models and then optimizes over this grid, to calculate the maximum likelihood estimator of the transformation.

Usage

```
boxcoxtype(formula, random = ~1, k = 3, trials = 1, data,
  find.in.range = c(-2, 2), s = 20, plot.opt = 1,
  random.distribution = "np", ...)

boxcoxpower(Lambda = 0)

binomial(link = boxcoxpower(0))
```

Arguments

formula	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
random	a formula defining the random model. Set random= ~1 to model logistic-type overdispersion model. For a two-level logistic-type model, set random= ~1|groups, where groups are at the upper level.
k	the number of mass points.
trials	optional prior weights for the data. For Bernoulli distribution, set trials=1.
data	a data frame containing variables used in the fixed and random effect models.
find.in.range	search in a range of Lambda, with default (-2,2) in step of 0.1.
s	number of points in the grid search of Lambda.
plot.opt	Set plot.opt=1, to plot the profile log-likelihood against Lambda. if plot.opt=0, no plot is printed.
random.distribution	the mixing distribution, Gaussian Quadrature (gq) or NPML (np) can be set.
...	extra arguments will be ignored.
Lambda	the power of the transformation
link	the link function to be used.

Details

The Box-Cox transformation (Box & Cox, 1964) is applied to the logistic mixed-effects models with an unspecified mixing distribution. The NPML estimate of the mixing distribution is known to be a discrete distribution involving a finite number of mass-points and corresponding masses (Aitkin et al., 2009). An Expectation-Maximization (EM) algorithm is used for fitting the finite mixture distribution, one needs to specify the number of components k of the finite mixture in advance. This algorithm can be implemented using the npmlreg function alldist for the logistic-type overdispersion model and the npmlreg function allvc for the two-level logistic-type model, setting family = binomial(link = boxcoxpower(Lambda)) where Lambda is the value of the power transformation. When k=1, the npmlreg function alldist() fits the logistic regression model without random effects.

boxcoxtype() performs a grid search over the parameter Lambda and then optimizes over this grid, to calculate the maximum likelihood estimator of the transformation. It produces a plot of the profile likelihood function that summarises information concerning Lambda, including a vertical line indicating the best value of Lambda that maximizes the profile log-likelihood.

Value

`Maximum`	the best estimate of `Lambda` found.
`objective`	the value of the profile log-likelihood corresponding to Maximum.
`coef`	the vector of coefficients.
`profile.loglik`	the profile log-likelihood of the fitted regression model.
`fit`	the fitted alldist object from the last EM iteration.
`aic`	the Akaike information criterion of the fitted regression model.
`bic`	the Bayesian information criterion of the fitted regression model.

The other outcomes are not relevant to users and they are intended for internal use only.

Author(s)

Amani Almohaimeed and Jochen Einbeck

References

Box G. and Cox D. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), pages 211-252.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. (2009). Statistical modelling in R. Oxford University Press Oxford.

Jochen Einbeck, Ross Darnell and John Hinde (2014). npmlreg: Nonparametric maximum likelihood estimation for random effect models. R package version 0.46-1.

See Also

`np.boxcoxmix`, `optim.boxcox`, `tolfind.boxcox`, `Kfind.boxcox`.

Examples

```
#Beta blockers data
data("betablocker", package = "flexmix")
library(npmlreg)
betavc <-allvc(cbind(Deaths, Total - Deaths) ~ Treatment, data = betablocker,random=~1|Center,
 k=3,random.distribution='np',family = binomial(link = boxcoxpower(0)))
betavc$disparity
#[1] 318.7211
betavc3 <-boxcoxtype(cbind(Deaths, Total - Deaths) ~ Treatment,random=~1|Center,
data = betablocker, find.in.range = c(-2,0.4),s=40,k=3,random.distribution='np')
#Maximum Profile Log-likelihood: -158.6025 at lambda= -0.56
betavc3$fit$disparity
#[1] 317.2049
betavc3$aic
#[1] 331.2049
betavc3$bic
#[1] 343.6942
```

`Kfind.boxcox` *Grid search over K for NPML estimation of random effect and variance component models*

Description

A grid search over the parameter `K`, to set the best number of mass-points.

Usage

```
Kfind.boxcox(formula, groups = 1, data, lambda = 1, EMdev.change = 1e-04,
  steps = 500, find.k = c(2, 10), model.selection = "aic", start = "gq",
  find.tol = c(0, 1.5), steps.tol = 15, ...)
```

Arguments

`formula`	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
`groups`	the random effects. To fit overdispersion models , set `groups = 1`.
`data`	a data frame containing variables used in the fixed and random effect models.
`lambda`	a transformation parameter, setting `lambda=1` means 'no transformation'.
`EMdev.change`	a small scalar, with default 0.0001, used to determine when to stop EM algorithm.
`steps`	maximum number of iterations for the EM algorithm.
`find.k`	search in a range of `K`, with default (2,10) in step of 1.
`model.selection`	Set `model.selection="aic"`, to use Akaike information criterion as model selection criterion or `model.selection="bic"`, to use Bayesian information criterion as model selection criterion.
`start`	a description of the initial values to be used in the fitted model, Quantile-based version "quantile" or Gaussian Quadrature "gq" can be set.
`find.tol`	search in a range of `tol`, with default (0,1.5) in step of 1.
`steps.tol`	number of points in the grid search of `tol`.
`...`	extra arguments will be ignored.

Details

Not only the shape of the distribution causes the skewness it may due to the use of an insufficient number of classes, `K`. For this, the `Kfind.boxcox()` function was created to search over a selected range of `K` and find the best. For each number of classes, a grid search over `tol` is performed and the `tol` with the lowest `aic` or `bic` value is considered as the optimal. Having the minimal `aic` or `bic` values for a whole range of `K` that have been specified beforehand, the `Kfind.boxcox()` function can find the best number of the component as the one with the smallest value. It also plots the `aic` or `bic` values against the selected range of `K`, including a vertical line indicating the best value of `K` that minimizes the model selection criteria. The full range of classes and their corresponding optimal `tol` can be printed off from the `Kfind.boxcox()`'s output and used with other **boxcoxmix** functions as starting points.

Value

`MinDisparity`	the minimum disparity found.
`Best.K`	the value of `K` corresponding to `MinDisparity`.
`AllMinDisparities`	a vector containing all minimum disparities calculated on the grid.
`AllMintol`	list of `tol` values used in the grid.
`All.K`	list of `K` values used in the grid.
`All.aic`	the Akaike information criterion of all fitted regression models.
`All.bic`	the Bayesian information criterion of all fitted regression models.

Author(s)

Amani Almohaimeed and Jochen Einbeck

See Also

`tolfind.boxcox`.

Examples

```
# Fabric data
data(fabric, package = "npmlreg")
teststr<-Kfind.boxcox(y ~ x, data = fabric,  start = "gq", groups=1,
find.k = c(2, 3), model.selection = "aic", steps.tol=5)
# Minimal AIC: 202.2114 at K= 2
```

`np.boxcoxmix` *Response Transformations for Random Effect and Variance Component Models*

Description

The function `np.boxcoxmix()` fits an overdispersed generalized linear model and variance component models using nonparametric profile maximum likelihood.

Usage

```
np.boxcoxmix(formula, groups = 1, data, K = 3, tol = 0.5, lambda = 1,
  steps = 500, EMdev.change = 1e-04, plot.opt = 1, verbose = TRUE,
  start = "gq", ...)
```

Arguments

formula	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
groups	the random effects. To fit overdispersion models , set groups = 1.
data	a data frame containing variables used in the fixed and random effect models.
K	the number of mass points.
tol	a positive scalar (usually, 0< tol <= 2)
lambda	a transformation parameter, setting lambda=1 means 'no transformation'.
steps	maximum number of iterations for the EM algorithm.
EMdev.change	a small scalar, with default 0.0001, used to determine when to stop EM algorithm.
plot.opt	Set plot.opt=1, to plot the disparity against iteration number. Use plot.opt=2 for tolfind.boxcox() and plot.opt=3 for optim.boxcox().
verbose	If set to FALSE, no printed output on progress.
start	a description of the initial values to be used in the fitted model, Quantile-based version "quantile" or Gaussian Quadrature "gq" can be set.
...	extra arguments will be ignored.

Details

The Box-Cox transformation (Box & Cox, 1964) is applied to the overdispersed generalized linear models and variance component models with an unspecified mixing distribution. The NPML estimate of the mixing distribution is known to be a discrete distribution involving a finite number of mass-points and corresponding masses (Aitkin et al., 2009). An Expectation-Maximization (EM) algorithm is used for fitting the finite mixture distribution, one needs to specify the number of components K of the finite mixture in advance. To stop the EM-algorithm when it reached its convergence point, we need to defined the convergence criteria that is the absolute change in the successive log-likelihood function values being less than an arbitrary parameter such as EMdev.change = 0.0001 (Einbeck et at., 2014). This algorithm can be implemented using the function np.boxcoxmix(), which is designed to account for overdispersed generalized linear models and variance component models using the non-parametric profile maximum likelihood (NPPML) estimation.

The ability of the EM algorithm to locate the global maximum in fewer iterations can be affected by the choice of initial values, the function np.boxcoxmix() allows us to choose from two different methods to set the initial value of the mass points. When option "gq" is set, then Gauss-Hermite masses and mass points are used as starting points in the EM algorithm, while setting start= "quantile" uses the Quantile-based version to select the starting points.

Value

mass.point	the fitted mass points.
p	the masses corresponding to the mixing proportions.
beta	the vector of coefficients.
sigma	the standard deviation of the mixing distribution (the square root of the variance).
se	the standard error of the estimate.
w	a matrix of posterior probabilities that element i comes from cluster k.

`loglik`	the log-likelihood of the fitted regression model.
`complete.loglik`	the complete log-likelihood of the fitted regression model.
`disparity`	the disparity of the fitted regression model.
`EMiteration`	provides the number of iterations of the EM algorithm.
`EMconverged`	TRUE means the EM algorithm converged.
`call`	the matched call.
`formula`	the formula provided.
`data`	the data argument.
`aic`	the Akaike information criterion of the fitted regression model.
`bic`	the Bayesian information criterion of the fitted regression model.
`fitted`	the fitted values for the individual observations.
`fitted.transformed`	the fitted values for the individual transformed observations.
`residuals`	the difference between the observed values and the fitted values.
`residuals.transformed`	the difference between the transformed observed values and the transformed fitted values.
`predicted.re`	a vector of predicted residuals.

The other outcomes are not relevant to users and they are intended for internal use only.

Author(s)

Amani Almohaimeed and Jochen Einbeck

References

Box G. and Cox D. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), pages 211-252.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. (2009). Statistical modelling in R. Oxford University Press Oxford.

Jochen Einbeck, Ross Darnell and John Hinde (2014). npmlreg: Nonparametric maximum likelihood estimation for random effect models. R package version 0.46-1.

See Also

`optim.boxcox`, `tolfind.boxcox`.

Examples

```
#Pennsylvanian Hospital Stay Data
data(hosp, package = "npmlreg")
test1 <- np.boxcoxmix(duration ~ age + wbc1, data = hosp, K = 2, tol = 1,
    start = "quantile", lambda = 1)
round(summary(test1)$w, digits = 3)
# [1,] 1.000 0.000

# Refinery yield of gasoline Data
data(Gasoline, package = "nlme")
test2.vc <- np.boxcoxmix(yield ~ endpoint + vapor, groups = Gasoline$Sample,
```

```
      data = Gasoline, K = 3, tol = 1.7, start = "quantile", lambda = 0)
test2.vc$disparity
# [1] 176.9827
```

---

np.estep *Internal boxcoxmix functions*

Description

auxiliary functions are not intended to be directly called from the user.

Usage

```
np.estep(y, x, lambda, p, beta, z, sigma)

np.zk(y, x, w, beta, lambda)

fik(y, x, lambda, beta, z, sigma)

np.theta(y, x, lambda, beta, z)

yhat(v, lambda = 1)

ytrans(y, lambda = 1)

np.bhat(y, x, w, z, lambda)

np.mstep(y, x, beta, lambda, w)

np.em(y, x, K, lambda = 1, steps = 500, tol = 0.5, start = "gq",
  EMdev.change = 1e-04, plot.opt = 1, verbose = TRUE, ...)

np.boxcox(formula, groups = 1, data, K = 3, tol = 0.5, lambda = 1,
  steps = 500, EMdev.change = 1e-04, plot.opt = 1, verbose = TRUE,
  start = "gq", ...)

vc.estep(Y, X, sizes = 1, lambda, p, beta, z, sigma)

zk(Y, X, sizes, w, beta, lambda)

bhat(Y, X, sizes, w, z, lambda)

mik(Y, X, sizes, lambda, beta, z, sigma)
```

```
vc.theta(Y, X, sizes, lambda, beta, z)

vc.mstep(Y, X, sizes = 1, beta, lambda, w)

vc.em(y, x, sizes = 1, K, lambda, steps = 500, tol = 0.5, start = "gq",
  EMdev.change = 1e-04, plot.opt = 1, verbose = TRUE, ...)

vc.boxcox(formula, groups = 1, data, K = 3, tol = 0.5, lambda = 1,
  steps = 500, EMdev.change = 1e-04, plot.opt = 1, verbose = TRUE,
  start = "gq", ...)

gqz(numnodes = 20, minweight = 1e-06)

masspoint.class(object)
```

Arguments

y	..
x	..
lambda	a transformation parameter, setting lambda=1 means 'no transformation'.
p	..
beta	..
z	..
sigma	..
w	..
v	..
K	the number of mass points.
steps	maximum number of iterations for the EM algorithm.
tol	a positive scalar (usually, 0< tol <= 2)
start	a description of the initial values to be used in the fitted model, Quantile-based version "quantile" or Gaussian Quadrature "gq" can be set.
EMdev.change	a small scalar, with default 0.0001, used to determine when to stop EM algorithm.
plot.opt	Set plot.opt=1, to plot the disparity against iteration number. Use plot.opt=2 for tolfind.boxcox and plot.opt=3 for optim.boxcox.
verbose	If set to FALSE, no printed output on progress.
...	extra arguments will be ignored.
formula	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
groups	the random effects. To fit overdispersion models , set groups = 1.
data	a data frame containing variables used in the fixed and random effect models.
Y	..
X	..
sizes	..
numnodes	..
minweight	..
object	..

Details

Internal boxcoxmix functions

Author(s)

Amani Almohaimeed and Jochen Einbeck

`optim.boxcox` *Response Transformations for Random Effect and Variance Component Models*

Description

The `optim.boxcox()` performs a grid search over the parameter `lambda` for overdispersed generalized linear models and variance component models and then optimizes over this grid, to calculate the maximum likelihood estimator of the transformation.

Usage

```
optim.boxcox(formula, groups = 1, data, K = 3, steps = 500, tol = 0.5,
  start = "gq", EMdev.change = 1e-04, find.in.range = c(-3, 3), s = 60,
  plot.opt = 3, verbose = FALSE, noformat = FALSE, ...)
```

Arguments

`formula`	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
`groups`	the random effects. To fit overdispersion models, set `groups = 1`.
`data`	a data frame containing variables used in the fixed and random effect models.
`K`	the number of mass points.
`steps`	maximum number of iterations for the EM algorithm.
`tol`	a positive scalar (usually, 0<`tol` <= 2)
`start`	a description of the initial values to be used in the fitted model, Quantile-based version "quantile" or Gaussian Quadrature "gq" can be set.
`EMdev.change`	a small scalar, with default 0.0001, used to determine when to stop EM algorithm.
`find.in.range`	search in a range of `lambda`, with default (-3,3) in step of 0.1.
`s`	number of points in the grid search of `lambda`.
`plot.opt`	Set `plot.opt=3`, to plot the disparity against iteration number and the profile log-likelihood against `lambda`. Use `plot.opt=0`, to only plot the profile log-likelihood against `lambda`.
`verbose`	If set to FALSE, no printed output on progress.
`noformat`	Set `noformat = TRUE`, to change the formatting of the plots.
`...`	extra arguments will be ignored.

Details

The Box-Cox transformation (Box & Cox, 1964) is applied to the overdispersed generalized linear models and variance component models with an unspecified mixing distribution. The NPML estimate of the mixing distribution is known to be a discrete distribution involving a finite number of mass-points and corresponding masses (Aitkin et al., 2009). An Expectation-Maximization (EM) algorithm is used for fitting the finite mixture distribution, one needs to specify the number of components `K` of the finite mixture in advance. To stop the EM-algorithm when it reached its convergence point, we need to defined the convergence criteria that is the absolute change in the successive log-likelihood function values being less than an arbitrary parameter such as `EMdev.change` = 0.0001 (Einbeck et at., 2014). This algorithm can be implemented using the function `np.boxcoxmix()`, which is designed to account for overdispersed generalized linear models and variance component models using the non-parametric profile maximum likelihood (NPPML) estimation.

The ability of the EM algorithm to locate the global maximum in fewer iterations can be affected by the choice of initial values, the function `optim.boxcox()` allows us to choose from two different methods to set the initial value of the mass points. When option "gq" is set, then Gauss-Hermite masses and mass points are used as starting points in the EM algorithm, while setting start= "quantile" uses the Quantile-based version to select the starting points.

`optim.boxcox()` performs a grid search over the parameter `lambda` and then optimizes over this grid, to calculate the maximum likelihood estimator of the transformation. It produces a plot of the non-parametric profile likelihood function that summarises information concerning `lambda`, including a vertical line indicating the best value of `lambda` that maximizes the non-parametric profile log-likelihood.

Value

`All.lambda`	list of `lambda` values used in the grid.
`Maximum`	the best estimate of `lambda` found.
`objective`	the value of the profile log-likelihood corresponding to Maximum.
`EMconverged`	1 is TRUE, means the EM algorithm converged.
`EMiteration`	provides the number of iterations of the EM algorithm.
`mass.point`	the fitted mass points.
`p`	the masses corresponding to the mixing proportions.
`beta`	the vector of coefficients.
`sigma`	the standard deviation of the mixing distribution (the square root of the variance).
`se`	the standard error of the estimate.
`w`	a matrix of posterior probabilities that element i comes from cluster k.
`loglik`	the profile log-likelihood of the fitted regression model.
`profile.loglik`	the profile complete log-likelihood of the fitted regression model.
`disparity`	the disparity of the fitted regression model.
`call`	the matched call.
`formula`	the formula provided.
`data`	the data argument.
`aic`	the Akaike information criterion of the fitted regression model.
`fitted`	the fitted values for the individual observations.

`fitted.transformed`
: the fitted values for the individual transformed observations.

`residuals`
: the difference between the observed values and the fitted values.

`residuals.transformed`
: the difference between the transformed observed values and the transformed fitted values.

`predicted.re`
: a vector of predicted residuals.

The other outcomes are not relevant to users and they are intended for internal use only.

Author(s)

Amani Almohaimeed and Jochen Einbeck

References

Box G. and Cox D. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), pages 211-252.

Aitkin, M. A., Francis, B., Hinde, J., and Darnell, R. (2009). Statistical modelling in R. Oxford University Press Oxford.

Jochen Einbeck, Ross Darnell and John Hinde (2014). npmlreg: Nonparametric maximum likelihood estimation for random effect models. R package version 0.46-1.

See Also

`np.boxcoxmix`, `tolfind.boxcox`.

Examples

```
# The strength Data
data(strength, package = "mdscore")
maxlam <- optim.boxcox(y ~ cut*lot, data = strength, K = 3,
           start = "gq" ,  find.in.range = c(-2, 2), s = 5)
# Maximum profile log-likelihood: 33.6795 at lambda= -0.4

data(Oxboys, package = "nlme")
Oxboys$boy <- gl(26,9)
maxlamvc <- optim.boxcox(height ~  age, groups = Oxboys$boy,
                         data = Oxboys,   K = 2,  start = "gq",
                         find.in.range=c(-1.2,1), s=6, plot.opt = 0)
maxlamvc$Maximum
#[1] -0.8333333
plot(maxlamvc,8)
```

plot *Plot diagnostics for boxcoxmix functions*

Description

plot() is a generic function used to produce some useful diagnostic plotting of the functions: np.boxcoxmix(), optim.boxcox() and tolfind.boxcox().

Usage

```
## S3 method for class 'boxcoxmix'
plot(x, plot.opt = 1, ...)
```

Arguments

x	an object for which a plot is desired.
plot.opt	an integer value between 1 and 8.
...	additional arguments.

Details

Plot diagnostics for boxcoxmix functions

Value

The plots to be printed depend on the number given in plot.opt, for the np.boxcoxmix(), optim.boxcox() and tolfind.boxcox() functions:

1	the disparities with the iteration number against the mass points
2	the fitted value against the response of the original and the transformed Data.
3	probability plot of residuals of the original against the transformed data.
4	individual posterior probabilities.
5	control charts of residuals of the original against the transformed data.
6	The histograms of residuals of the original against the transformed data.
7	works only for the tolfind.boxcox() function and plots the specified range of tol against the disparities
8	works only for the optim.boxcox() function and gives the profile likelihood function that summarises information concerning lambda.
9	works only for the Kfind.boxcox() function and plots the specified range of K against the AIC or BIC information criteria
10	works only for the boxcoxtype() function and gives the profile likelihood function that summarises information concerning lambda for generalized linear Mixed-effects Models.

print.boxcoxmix *Summary of boxcoxmix functions*

Description

summary() and print() are generic functions used to produce the results of the functions: np.boxcoxmix(), optim.boxcox() and tolfind.boxcox().

Usage

```
## S3 method for class 'boxcoxmix'
print(x, digits = max(3, getOption("digits") - 3),
  na.print = "", ...)

## S3 method for class 'boxcoxmixpure'
print(x, digits = max(3, getOption("digits") - 3),
  na.print = "", ...)

## S3 method for class 'boxcoxmix'
summary(object, digits = max(3, getOption("digits") - 3),
  ...)

## S3 method for class 'boxcoxmixpure'
summary(object, digits = max(3, getOption("digits") -
  3), ...)
```

Arguments

x	an object for which a summary is desired.
digits	an integer number format.
na.print	a character string which is used to indicate NA values output format.
...	additional arguments.
object	an object for which a summary is desired.

Details

Summary of boxcoxmix functions

tolfind.boxcox *Grid search over tol for NPPML estimation of random effect and variance component models*

Description

A grid search over the parameter tol, to set the initial values of the EM algorithm.

Usage

```
tolfind.boxcox(formula, groups = 1, data, K = 3, lambda = 1,
  EMdev.change = 1e-04, plot.opt = 2, s = 15, steps = 500,
  find.in.range = c(0, 1.5), start = "gq", verbose = FALSE,
  noformat = FALSE, ...)
```

Arguments

formula	a formula describing the transformed response and the fixed effect model (e.g. y ~ x).
groups	the random effects. To fit overdispersion models , set groups = 1.
data	a data frame containing variables used in the fixed and random effect models.
K	the number of mass points.
lambda	a transformation parameter, setting lambda=1 means 'no transformation'.
EMdev.change	a small scalar, with default 0.0001, used to determine when to stop EM algorithm.
plot.opt	Set plot.opt=2, to plot the EM trajectories and the development of the disparity over iteration number. And plot.opt=0, for none of them.
s	number of points in the grid search of tol.
steps	maximum number of iterations for the EM algorithm.
find.in.range	search in a range of tol, with default (0,1.5) in step of 0.1 .
start	a description of the initial values to be used in the fitted model, Quantile-based version "quantile" or Gaussian Quadrature "gq" can be set.
verbose	If set to FALSE, no printed output on progress.
noformat	Set noformat = TRUE, to change the formatting of the plots.
...	extra arguments will be ignored.

Details

A grid search over tol can be performed using tolfind.boxcox() function, which works for np.boxcoxmix() to find the optimal solution.

Value

MinDisparity	the minimum disparity found.
Mintol	the value of tol corresponding to MinDisparity.
AllDisparities	a vector containing all disparities calculated on the grid.
Alltol	list of tol values used in the grid.
AllEMconverged	1 is TRUE, means the EM algorithm converged.
aic	the Akaike information criterion of the fitted regression model.
bic	the Bayesian information criterion of the fitted regression model.

Author(s)

Amani Almohaimeed and Jochen Einbeck

See Also

np.boxcoxmix.

Examples

```
# The Pennsylvanian Hospital Stay Data
data(hosp, package = "npmlreg")
test1 <- tolfind.boxcox(duration ~ age , data = hosp, K = 2, lambda = 0,
           find.in.range = c(0, 2), s = 10,  start = "gq")
# Minimal Disparity: 137.8368 at tol= 2
# Minimal Disparity with EM converged: 137.8368 at tol= 2

# Effect of Phenylbiguanide on Blood Pressure
data(PBG, package = "nlme")
test2 <- tolfind.boxcox(deltaBP ~ dose , groups = PBG$Rabbit, find.in.range = c(0, 2),
    data = PBG, K = 2, lambda = -1, s = 15,  start = "quantile", plot.opt = 0)
test2$Mintol
# [1] 1.6
test2$MinDisparity
# [1] 449.5876
```

Index